NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

SOLVING THE MULTICOMMODITY TRANSSHIPMENT PROBLEM

by

Cyrus James Staniec

June 1987

Thesis Advisor: Gerald G. Brown

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 55	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
SOLVING THE MULTICOMMODITY TRANSSHIPMENT PROBLEM

12 PERSONAL AUTHOR(S)
Staniec, Cyrus J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Doctoral Dissertation	FROM ______ TO ______	1987, June	126

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Multicommodity Transshipment Problem; Large-Scale Optimization; Price Direction; Resource Direction; Dual Decomposition; Restricted

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

We examine two categories of solution algorithms for the large-scale multicommodity transshipment problem (MCTP): resource direction and price direction. In the former category we construct RDLB, a new algorithm which uses a simplified projection method in the subgradient capacity reallocations and conjugate subgradient directions with approximate line search to provide better termination conditions in the Lagrangean lower-bounding iteration. In the latter category, we develop DDC, a dual decomposition, and we introduce RSD(P) and RSD(A), new non-linear decomposition algorithms for the MCTP based on penalty transformations of the original problem and using restricted simplicial decomposition.

Computational results are presented for four- and ten-product versions of a problem with an underlying network of 3,300 nodes and 10,400 arcs.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT: [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: Prof. Gerald G. Brown	22b TELEPHONE (Include Area Code): (408) 646-2140; 22c OFFICE SYMBOL: Code 55Bw

#18 - SUBJECT TERMS (CONTINUED)

Simplicial Decomposition; Subgradient Optimization; Penalty Functions; Augmented Lagrangean

#19 - ABSTRACT (CONTINUED)

Results show RDLB stalls before reaching optimality, apparently a common problem in primal subgradient reallocations, while the RSD algorithms reach near-optimal solutions up to 10 times faster than a direct primal simplex-based solver, and display very favorable convergence rates compared to DDC. As a final test, RSD(A) and DDC are applied to a 100-product problem totaling 330,000 nodes and 1,040,000 arcs. RSD(A) reaches an acceptable solution within 4% of optimality in under 17 minutes, while DDC terminates after 68 minutes with a 12% gap remaining around the optimal solution.

Solving the Multicommodity Transshipment Problem

by

Cyrus James Staniec
Major, United States Army
B.S.Ch.E., Syracuse University, 1971
M.S., Naval Postgraduate School, 1984

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 1987

ABSTRACT

We examine two categories of solution algorithms for the large-scale multicommodity transshipment problem (MCTP): resource direction and price direction. In the former category we construct RDLB, a new algorithm which uses a simplified projection method in the subgradient capacity reallocations and conjugate subgradient directions with approximate line search to provide better termination conditions in the Lagrangean lower-bounding iteration. In the latter category, we develop DDC, a dual decomposition, and we introduce RSD(P) and RSD(A), new non-linear decomposition algorithms for the MCTP based on penalty transformations of the original problem and using restricted simplicial decomposition.

Computational results are presented for four- and ten-product versions of a problem with an underlying network of 3,300 nodes and 10,400 arcs. Results show RDLB stalls before reaching optimality, apparently a common problem in primal subgradient reallocations, while the RSD algorithms reach near-optimal solutions up to 10 times faster than a direct primal simplex-based solver, and display very favorable convergence rates compared to DDC. As a final test, RSD(A) and DDC are applied to a 100-product problem totaling 330,000 nodes and 1,040,000 arcs. RSD(A) reaches

an acceptable solution within 4% of optimality in under 17 minutes, while DDC terminates after 68 minutes with a 12% gap remaining around the optimal solution.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I extend my sincere appreciation to those who have significantly influenced this research effort, namely:

Rick Rosenthal, who guided my earliest efforts in large-scale MCTP's,

Kevin Wood, whose shotgun-blasts of insight and inquisition left me bloodied but with deeper understanding, and most especially to

Gerry Brown, an able mentor for such a foray through n-space, and an admirable human being.

I also thank those whose support was vital in bringing this all together;

Bob Lande, my able, patient, and timely typist,

Patrick F. McCrary of Chevron Corporation who kindly ran our "dogs" for us, and

David F. Norman, Manager of Systems and Operations at the W.R. Church Computer Center whose expertise was vital in beating the machines into submission.

Lastly, but most importantly, I thank my daughters, Deanna and Jilian, for their patience and understanding during this effort, and thank my wonderful wife, Donna, who tended the family while I was out to lunch.

To God be the Glory.

I. INTRODUCTION

Minimum-cost single-commodity capacitated transshipment problems are now solved routinely using primal network simplex codes. See, for instance, (Bradley, Brown, and Graves, 1977). However, when multiple products flow over the same network, the pure network structure can be confounded by the presence of constraints limiting the total flow of all products on each arc. Thus, single-commodity solvers may not be applied directly, and, as the number of products increases, the size of the constraint matrix grows so rapidly that conventional simplex solvers become useless.

Because it is a frequently encountered problem, specialized algorithms for the multicommodity transshipment problem (MCTP) have been widely studied and reported in the literature. However, none of the methods has been so generally successful that it dominates other methods, as primal network algorithms do in the single-commodity arena.

This paper documents new algorithms for the MCTP which fall into the broad categories of "resource-directive" and "price-directive." The first algorithm is an enhancement of a popular resource-directive approach using subgradient optimization, but incorporating a simplified primal projection together with conjugate subgradient directions in the Lagrangean lower bound steps. The algorithm promises ease of computation in the primal restriction and improved criteria for termination over previous algorithms, but apparently shares a common problem of subgradient-based resource-direction: stalling in the primal before reaching optimality.

The set of price-directive algorithms we present includes a variant of dual (Dantzig-Wolfe) decomposition and a new family of decomposition algorithms using a penalty transformation of the original MCTP to create a non-linear master problem which is solved by restricted simplicial decomposition. This is a unique approach to the MCTP not previously attempted which exhibits superior convergence rates in computational tests compared to other solution methods. The algorithm is quite general and therefore applicable to a wide variety of linear programs exhibiting complicating constraints.

We introduce the MCTP in the following section, and then give an overview of major solution approaches to the MCTP in Section B, and a more detailed literature review in Section C. The resource-directive algorithm is presented in Chapter II and the price-directive algorithms in Chapter III. Computational results appear in Chapter IV and Chapter V presents conclusions and areas for future research.

A. STATEMENT OF THE PROBLEM

The general form of linear program in which we are interested is:

(LP)	$\min \; cx$	(duals)
	st $\; A_1x \leq b_1$	(u_1)
	$A_2x = b_2$	(u_2)
	$0 \leq x \leq b_3$	

where one set of constraints, $A_2x = b_2$, $0 \leq x \leq b_3$ is deemed to be easy to solve, and a second set, $A_1x \leq b_1$, complicates the problem. The vector $u = (u_1,u_2)$ is the set of dual multipliers associated with the constraints with $u_1 \leq 0$ and u_2 unrestricted in sign.

The following notation will be used throughout this presentation. We assume all products flow over the same underlying network. Let $|Z|$ denote the cardinality of a set Z.

$T = \{I,J\}$	is a transshipment network with a set of nodes, I, and a set of arcs, J.
P	is the set of products flowing on T.
$i \in I$	is a node of the network.
$j \in J$	is an arc in the network.
$p \in P$	is a product using network T.
N_p	is an $(\|I\| \times \|J\|)$ node-arc incidence matrix for each product $(N_1 = \ldots = N_{\|P\|})$.
N	is an $(\|I\| \cdot \|P\|) \times (\|J\| \cdot \|P\|)$ matrix with the N_p matrices along the diagonal, 0's elsewhere.

$c = \{c_1, \ldots, c_p, \ldots, c_{|P|}\}$ is a vector of costs on the arcs, length $|J| \cdot |P|$.

$x = \{x_1, \ldots, x_p, \ldots, x_{|P|}\}$ is a vector of flows on the arcs, length $|J| \cdot |P|$.

b_1 is a vector of joint capacities with length $|J|$

b_2 = right hand side with

$b_{2pi} > 0$ if product p has a supply at node i,

$b_{2pi} < 0$ if product p has a demand at node i, and

$b_{2pi} = 0$ otherwise.

A is a $|J| \times |J| \cdot |P|$ matrix = $\{I, \ldots, I\}$.

We specialize LP to MCTP by letting $A_2 = N$, $b_3 = b_1$, and dropping the subscript on A_1:

(MCTP)	min	cx	(duals)	(1.1)
	st	$Ax \leq b_1$	(u_1)	(1.2)
		$Nx = b_2$	(u_2)	(1.3)
		$0 \leq x_p \leq b_1$ for all $p \in P$.		(1.4)

The easy constraints are the single-product pure network flow constraints (1.3,1.4), and the complicating constraints are the joint capacitation constraints (1.2). We assume for notational simplicity that all arcs have joint capacities, although in practice only a subset have such restrictions.

For convenience, let

$$F = \{x | Nx = b_2,\ 0 \leq x_p \leq b_1 \text{ for all } p \in P\}$$

be the set of all feasible single-commodity flows with joint capacity constraints relaxed. In the absence of the complicating these constraints (1.2), the MCTP decouples into a set of independent single commodity networks.

The Lagrangean dual of MCTP with respect to the joint capacity constraints is found by placing these constraints in the objective with multipliers $u_1 \leq 0$:

$$(LR) \qquad \max_{u_1 \leq 0} \min_{x \geq 0} L(u_1, x) = cx - u_1(Ax - b_1)$$
$$\text{st} \quad x \in F$$

According to duality theory, if x* solves MCTP, then $L(u_1,x) \leq cx^*$ for $u_1 \leq 0$ and $x \in F$. Furthermore, a solution (u_1^{**}, x^{**}) to LR has $L(u_1^{**}, x^{**}) = cx^*$. Thus, we may use LR to generate a lower bound on MCTP by fixing $u_1 \leq 0$ and solving the following problem:

$$(LR(u_1)) \qquad \min_{x} cx - u_1(Ax - b_1)$$
$$\text{st} \quad x \in F ,$$

and this bound will be tight if u_1 is chosen correctly. Solving the Lagrangean dual generally allows some constraints to be violated with penalty $u_1(Ax-b_1)$. For any $x \in F$, we denote the set of violated constraints as

$$J_V = \{j \mid j \in J, A_j x > b_{1j}\} \quad . \qquad (1.5)$$

B. OVERVIEW OF SOLUTION TECHNIQUES

Solution methods for large-scale linear programs may be divided into three broad categories: direct factorization or compact inverse methods, indirect resource-directive methods, and indirect price-directive methods. Factorization approaches exploit specific structure inherent within the constraints to produce a compact basis representation with which the steps of a primal or dual simplex algorithm may be performed. Direct factorization is not pursued in this study (see, for instance, Graves and McBride, 1976).

The other two approaches, resource and price direction are decompositions which divide the original problem into a master problem and subproblems which exchange information to solve the original problem indirectly by iteration.

The resource-directive approach solves the MCTP as a two part minimization:

$$\begin{array}{lrl}
\min\limits_{y\geq 0}\ \min\limits_{x\geq 0} & cx & \\
\text{st} & Nx & = b_2 \\
 & x - y & \leq 0 \\
 & Ay & = b_1 \\
 & y_p & \leq b_1 \quad \text{for all } p \in P \ .
\end{array}$$

The vector y allocates joint capacities to the individual commodities. For fixed y, the solution to the inner minimization is

$$(\mathrm{RS}(y)) \qquad \bar{V}(y) = \min \; cx \quad \text{st} \quad Nx = b_2, \quad 0 \le x \le y$$

which is the "subproblem" or "subproblems" since the individual commodities are no longer coupled and may be solved independently.

The outer minimization can now be written as

$$(\mathrm{RD}) \qquad \min \; \bar{V}(y) \quad \text{st} \quad Ay = b_1, \quad y_p \le b_1 \text{ for all } p \in P, \quad y \ge 0 .$$

Standard methods for solving (RD) are Benders decomposition and subgradient optimization. Benders decomposition creates a master problem which makes successive tangential approximations to $\bar{V}(y)$; the tangent planes are derived by solving subproblems (RS(y)) whose capacity (resource) allocations are determined by the master problem. More details of Benders decomposition are given in Chapter III but the method is not pursued because the master problems become unwieldy.

Subgradient optimization solves (RD) in a fashion which is analogous to a projected gradient algorithm which could be used if $\bar{V}(y)$ were differentiable. Given a feasible allocation y^k in the kth iteration, a feasible reallocation is obtained by

$$y^{k+1} = y^k + s^k d^k$$

where s^k is a scalar step length and d_k is a feasible direction. Since $\overline{V}(y)$ is not everywhere differentiable, d^k is a projected subgradient rather than a projected gradient. Standard subgradient optimization does not use a line search to determine s^k since d^k is not guaranteed to be a descent direction. However, we devise an algorithm which performs an approximate line search when applicable in the lower bound routine. Subgradients are determined via solutions of the subproblems RS(y); the master problem of this procedure is the reallocation mechanism.

We present price direction as a class of penalty problems,

$$\begin{aligned} &\max_{z \geq 0} \min_{x} \quad cx + P(z, Ax-b_1) \\ &\text{st} \qquad\quad x \in F \end{aligned} \tag{1.6}$$

where $P(\cdot)$ is a penalty term involving those constraints deemed to be complicating. The vector z is determined differently for various forms of $P(\cdot)$, but in general is involved in constructing an optimal vector of dual multipliers.

Letting $P(u_1, Ax-b_1) = -u_1(Ax-b_1)$, we see that (1.6) becomes LR, $-z \equiv u_1$ estimates the optimal dual multipliers, and by fixing $u_1 \leq 0$, the inner minimization amounts to solving the Lagrangean subproblem LR(u_1).

Solution of the Lagrangean dual, i.e., (1.6) with linear penalties, cannot guarantee an optimal solution to MCTP although optimal u_1^* and x^* for MCTP are in the set of solutions to LR. Nevertheless, optimizing LR is useful for bounding purposes and it is sometimes possible to obtain good solutions to MCTP from LR. The method used in this work to solve LR is subgradient optimization.

Subgradient optimization for LR simply updates the multiplier estimates at each iteration k, by the formula

$$u_1^{k+1} = \min\ (0, u_1^k - s^k(Ax^k - b_1))$$

while controlling the scalar steplengths s^k, resolving the Lagrangean to obtain a new x^{k+1} and seeking feasibility only indirectly via the penalty terms. The dual update mechanism is the master problem for LR while the subproblems are $LR(u_1^k)$.

Standard dual (Dantzig-Wolfe) decomposition may be interpreted as a special method of solving (1.6) with piecewise-linear penalty function,

$$\begin{aligned} &P(z, Ax-b_1) = -z(Ax-b_1) \\ &\text{and } z_j = \infty \qquad \text{if } (Ax-b_1)_j > 0 \\ &\qquad z_j = 0 \qquad \text{if } (Ax-b_1)_j < 0 \\ &\qquad z_j \quad [0,\infty) \quad \text{if } (Ax-b_1)_j = 0\ . \end{aligned}$$

Z is determined by solving the master linear program of Chapter III.A, yielding the duals $u_1 = -z$. Subproblem solutions attempt to provide descent directions for this penalty function.

$P(\cdot)$ may also take standard non-linear forms such as the quadratic function, $.5h||Q(x)||^2$, where z is replaced here by the scalar h, $Q(x) = (A_j x - b_{1j})^+ = \max(0, A_j x - b_{1j})$, and $||\cdot||$ denotes the Euclidean norm. Penalty function theory tells us precisely how to solve this problem: solve the inner minimization as $h \to \infty$. As $h \to \infty$, $x \to x^*$ and $h(Ax-b_1)^+$ converges to an optimal set of dual multipliers.

However, starting with h large produces a highly ill-conditioned problem, so the problem is usually solved for a sequence of increasing values for h, producing a sequence of improving, but infeasible solutions. An augmented Lagrangean penalty function is investigated for reducing ill-conditioning and speeding convergence.

A reasonable approach for solving these nonlinear penalty problems is some feasible descent method. For fixed h, feasible descent directions are derived for $cx + P(h,Ax-b)$ at $x = \hat{x}$ by solving the linear subproblem

$$\begin{array}{ll} \min & \nabla_{\hat{x}}(c\hat{x} + P(h,A\hat{x}-b_1))x \\ \text{st} & \qquad\qquad\qquad x \in F \end{array}$$

to obtain $\bar{x}$. The Frank-Wolfe algorithm (1956) would perform a linear search in the direction $\bar{x}-\hat{x}$ to obtain a new x and iterate. This type of master problem tends to have poor convergence in practice so we employ Restricted Simplicial

Decomposition (Hearn et al., 1984) which maintains a restricted set of extreme points of F together with $\hat{x}$. Instead of solving a simple line search the master problem solves a nonlinear program over the convex hull of the retained points. The loss of simplicity in the master problem is typically offset by improved convergence of the overall algorithm.

There is no strong rationale for replacing a hard linear problem with a seemingly more difficult nonlinear problem. Consequently, this approach has not previously been considered for large-scale applications. However, we show in Chapter III that this approach can be attractive.

C. SURVEY OF RELATED LITERATURE

This section reviews the literature which has led to the current state of the art in algorithms for the MCTP. We begin with a brief overview of single-commodity network algorithms because of their computational importance in many algorithms for the MCTP. We then mention prior contributions to a few special cases of the MCTP, and finally review literature on algorithms used to solve general MCTPs.

Single-commodity network flow problems have been widely studied since the 1940s, for two primary reasons: they are frequently encountered and their special structure lends itself to algorithms which are more efficient than general linear programming techniques. The constraint matrix of the pure network problem is a node-arc incidence matrix: all 0's and ±1's, with at most one +1 and one -1 per column. This particular structure has three desirable properties: total unimodularity, which guarantees integer primal solutions given integer right-hand sides, and integer dual solutions given integer costs, a basis matrix which may be triangulated by permutation of rows and columns and thus simplifying computation, and primal extreme point solutions equivalent to spanning trees in the network. Several algorithms were developed in the 1950s and 1960s which made at least some use of these properties. Primal simplex methods were proposed by Dantzig (1963) and Fulkerson and

Dantzig (1955). Primal-dual methods became popular in practice at that time, including Kuhn's Hungarian Method (1955) and the out-of-kilter algorithm of Ford and Fulkerson (1962).

Primal-dual methods were favored throughout the 1960s, but some works, most notably the basis-labelling scheme of Johnson (1966), set the stage for research which led to the efficient primal network algorithms in use today. Subsequent research focused on efficient data structures and their manipulation, which led to compact basis representations and efficient performance of the simplex pivot. Significant research was done in the 1970s by Srinivasan and Thompson (1972,1973), Glover, Karney and Klingman (1974), Glover, Karney, Klingman and Napier (1974), Glover, Klingman and Stutz (1974), Barr, Glover, and Klingman (1979), and Bradley, Brown and Graves (1977). Results presented by these researchers demonstrated the primal network simplex to be up to two orders of magnitude faster, to require much less memory than general simplex solvers, and to be about 40% faster than out-of-kilter codes.

This research has given us network codes, such as GNET (Bradley, Brown and Graves, 1977), which can solve large (say, many thousands of nodes and arcs) single-commodity network problems very efficiently. This research is doubly important for the MCTP since it allows efficient subproblem solution in many of the algorithms developed for the MCTP.

Some of the pure network structure found in the single-commodity network problem carries over into the MCTP. Single-product networks do appear as blocks along the diagonal of the constraint matrix, but, unfortunately, the joint capacity constraints couple the networks together and generally destroy total unimodularity of the constraint matrix, admitting fractional solutions and requiring real arithmetic. It is appealing to try to restore total unimodularity or product independence by transformation. Some success in this area has been reported by Evans (1976,1978a,1978b,1983) and by Evans, Jarvis and Duke (1977). The class of problems for which their methods work is quite restricted, however: total unimodularity is shown to hold when the number of sources or sinks for each product is less than or equal to 2, and the existence of transformations to single-product networks is shown when the number of sinks per product equals 2. Thus, this is of little help to the general MCTP.

Algorithms for the general MCTP are broadly classified as direct methods which exploit special structure using the simplex algorithm or indirect methods which use some form of decomposition. The indirect methods include price-direction and resource-direction.

We first review the development of price-direction. Recognizing that the number of extreme points in a master problem may be huge, Ford and Fulkerson (1958) devised a procedure for the multicommodity maximal flow problem which uses column generation to produce favorable extreme points as needed. In their formulation of the problem, the variables correspond to chains or paths through the network, and the constraints represent individual arcs. Dantzig and Wolfe (1960) formalized the procedure into the dual decomposition procedure in which the master problem is solved to provide pricing information to the subproblems. Solving the subproblems produces an extreme point, which is added to the master problem if it is favorable, or indicates optimality of the current master problem solution if it is not.

Tomlin (1966) first formulated and implemented dual decomposition for the minimum cost MCTP, showing the equivalence of the Ford-Fulkerson algorithm to the algorithm described by Dantzig and Wolfe. Others have developed extensions of the MCTP using the dual-decomposition approach. Cremeans, Smith and Tyndall (1970) and Wollmer (1972) formulated a model in which flow on some arcs depends on the availability of resources which are shared with other arcs. Weigel and Cremeans (1972) extended the model to allow the flow of each commodity to be measured in distinct units, joint arc capacities in common units, and to incorporate node capacity constraints. Swoveland (1973) presented a generalization of the MCTP which involved a decomposition in which a single subproblem is solved in two stages.

Considered an effective technique in early iterations, dual decomposition has a reputation for poor convergence, with progress toward optimality tailing off in later iterations. Ho (1984a) speculated that this failure to converge is due to numerical error in computer implementation. Ho and Loute (1983) compared solution times of several problems for a commercial linear programming package and decomposition codes. They concluded that standard LP is more efficient when it is practical, but pointed out that decomposition is nearly as efficient for the large problems tested. Ho (1984b) further speculates that efficiency of the decomposition increases with problem dimension.

Simplicial decomposition, an indirect price-directive method applicable to non-linear quasi-convex programs was introduced by von Hohenbalken (1977). Convergence of a restricted version of simplicial decomposition was recently demonstrated by Hearn et al., (1984).

We now turn to resource-direction, which solves the MCTP using the subproblem

$$\text{RS}(y) \qquad \bar{V}(y) = \min \ cx$$
$$\text{st} \quad Nx = b_2$$
$$0 \le x \le y$$

and the master problem

$$\text{RD} \qquad \min \ \bar{V}(y)$$
$$\text{st} \quad Ay = b_1$$
$$y_p \le b_1 \quad \text{for all } p \in P$$
$$y \ge 0 \ .$$

Because the master problem provides allocations feasible in MCTP, RS(y) produces an upper bound on MCTP. When RS(y) has a feasible solution, it is feasible in MCTP. It is a simple matter for the modeller to introduce explicit artificial and slack arcs with penalty costs to insure that any allocation will produce a feasible solution to MCTP.

Resource-directive algorithms proceed by iteratively solving the restriction, and then updating the allocations in some favorable manner and resolving. Geoffrion (1970) provides a good overview of the resource-directive approach. The predominant method for computing new allocations is via subgradient directions, which are a generalization of the gradient for convex or concave functions with nondifferentiable points. This method is applicable since $\bar{V}(y)$ is a piece-wise linear, convex function.

The subgradient approach appeared first for solving sets of linear inequalities (Agmon, 1954; Motzkin and Schoenberg, 1954), and then appeared in the Russian literature adapted to optimization problems, e.g., Polyak (1967). The optimization version was first applied in the Western literature to the travelling salesman problem by Held and Karp (1970,1971), and further developed as a general method for optimization by Held, Wolfe and Crowder (1974). The convergence rate and stepsize considerations were studied by Goffin (1977) and Bazaraa and Sherali (1981). Fisher (1985) summarized the subgradient approach to solving the Lagrangean dual.

Resource-directive algorithms for the MCTP using subgradients in the direction-finding process for optimizing RD were developed by Kennington and Shalaby (1977), Rosenthal (1983), and Allen (1985). Ali, Helgason, Kennington, and Lall (1980) declare their resource-directive algorithm to be faster than either a price-directive or a special basis factorization code in their computational tests.

While subgradient resource-directive methods are considered exact (Kennington and Helgason, 1980), zigzagging in subgradient methods is known to be a problem (Sandi, 1979), so convergence to optimality is frequently slow at best. Thus, the reported computational advantage of resource direction using subgradient updates is based on its purported ability to reach an acceptable near-optimal point before simplex-based methods reach optimality (Ali et al., 1978, 1980; Kennington and Shalaby, 1977).

The final major category of algorithms for the MCTP is basis factorization or compact inverse methods. Algorithms in this category employ a primal or dual simplex approach, but exploit the special structure of the LP basis for the MCTP to reduce the size of the basis inverse that must be carried explicitly. A dual partitioning method was presented by Grigoriadis and White (1972). Graves and McBride (1976) discussed the primal approach in the general context of mathematical programming. Hartman and Lasdon (1972) presented a theoretical primal approach based on the Generalized Upper Bounding (GUB) technique of Dantzig and Van Slyke (1967). Incorporating graph theory, they factored the MCTP basis into a network basis for each product and a "working basis" with dimension equal to the number of currently binding joint capacity constraints. Several efficiencies result. Primal network simplex techniques apply to computation on pure network bases. Only computations involving the working basis require real arithmetic, and this basis is generally quite small.

Kennington (1977) implemented the method and further study has been done by Helgason and Kennington (1977). Ali, Barnett et al., (1984) concluded that the GUB approach is three to five times faster than standard LP codes, but other studies generally rate resource-directive algorithms superior to the GUB approach (see, for instance, Ali, et al., 1980, and Kennington and Shalaby, 1977).

II. A RESOURCE-DIRECTIVE APPROACH TO THE MCTP

This chapter presents a resource-directive method for solving the MCTP which incorporates a simplified projection mechanism in the primal subgradient capacity reallocations and conjugate subgradient directions with approximate line search in a Lagrangean lower-bounding routine. The procedure is based on the method of Allen (1985) which solves two essentially independent sequences of problems, the resource-directive sequence to generate feasible solutions and upper bounds, and a Lagrangean sequence to generate lower bounds. The purposes for these enhancements to Allen's procedure are to provide a computationally-supportable, theoretically sound projection in the reallocation routine, to provide a better mechanism for termination of the procedure, and to attempt to generate ascent directions in the Lagrangean routine to make line search worthwhile. We first restate the general form of the problem to be solved, and then introduce the concept and required theory of subgradient optimization. At that point we will be able to state clearly the shortcomings of previous approaches and present the procedure based on our improvements.

The resource directive approach attempts to solve the problem

$$\min_{y} \min_{x} \quad cx \tag{2.1}$$

$$\text{st} \quad Ax = b_1 \tag{2.2}$$

$$Nx = b_2 \tag{2.3}$$

$$0 \le y_p \le b_1, \quad \text{for all } p \in P \tag{2.4}$$

$$x - y \le 0 \tag{2.5}$$

$$x \ge 0 \; . \tag{2.6}$$

By selecting a particular $y \in Y$, i.e., a set of capacity allocations satisfying (2.2) and (2.4), the remaining subproblem is

$$\begin{array}{lll} \overline{V}(y) = \min cx & & \text{(duals)} \\ \quad \text{st} \quad Nx = b_2 & & (u_2) \\ \quad\quad\quad x \le y & & (u_3) \\ \quad\quad\quad x \ge 0 \;, & & \end{array}$$

which is a set of restricted, single-product transshipment problems. The original problem is then $\{\min \overline{V}(y) \text{ st } y \in Y\}$, so the outer minimization forms a master problem which seeks improving capacity allocations.

Before turning to the subgradient approach to solving this problem, we briefly mention another approach to this problem, "Benders decomposition." First, we make the simplifying assumption that all problems have finite feasible solutions and recall the theorem of duality to establish that the optimal values of a linear program and its dual are equal. Now the dual of our subproblem is

$$\begin{aligned} \bar{V}(y) = \max \quad & u_2b_2 - u_3y \\ \text{st} \quad & u_2N - u_3I \le c \\ & u_2 \text{ free}, u_3 \ge 0 , \end{aligned} \tag{2.7}$$

and letting the constraint set be represented by its extreme points, $\{u_{2e}\}$, $\{u_{3e}\}$, where $e \in E$, the index set of extreme points, it may be rewritten

$$\max_{e} \quad u_{2e}b_2 - u_{3e}y \ .$$

Substituting for $\bar{V}(y)$ in the master problem yields

$$\{\min \max_{e}(u_{2e}b_2 - u_{3e}y) \text{ st } y \in Y\} ,$$

or, equivalently,

$$\begin{aligned} \min \quad & z \\ \text{st} \quad & z \ge u_{2e}b_2 - u_{3e}y \quad e \in E \\ & Ay = b_1 \\ & 0 \le y \le b_1 \ . \end{aligned}$$

This is the Benders master problem. In practice, the extreme points of the subproblem are not all known, so we generate them by iteratively solving the subproblems to produce a new extreme point, and then add it to the master problem in the form of a constraint. The master, in turn, is solved to produce a new allocation. This is the process of the Benders (1962) decomposition algorithm which treats y as a complicating variable. A full presentation of this method, which also considers conditions of unboundedness, may be found in Lasdon (1970). The approach is not pursued here because of the large size of the Benders master problem.

The subgradient approach attempts to solve the master problem by simple updates of the capacity allocations at each iteration, $y^{k+1} = Pr[y^k + s^k d^k]$, where Pr indicates a projection such that $y^{k+1} \in Y$, d^k is a subgradient and s^k is from a scalar step sequence, $\{s^k\}$ satisfying

$$\sum s^k = \infty, \quad s^k \geq 0, \quad \text{and} \quad s^k \to 0 \text{ as } k \to \infty. \qquad (2.8)$$

Polyak (1967) demonstrated that these projected updates converge in the limit to an optimal solution simply by assuring that (2.8) is met. Letting $\hat{y}^{k+1} = y^k + s^k d^k$, the projection finds a feasible $y^{k+1} = \text{argmin}\{||y-\hat{y}^{k+1}||^2 \text{ st } y \in Y\}$.

The subgradient itself is a generalization of the gradient for a function with nondifferentiable points. If f is a convex function with nondifferentiable points defined on a feasible region F, then d is a subgradient of f at $\bar{z} \in F$ if

$$f(z) \geq f(\bar{z}) + d'(z-\bar{z}) \quad \text{for all } z \in F .$$

The set of subgradients at a point is called the subdifferential, defined by

$$\partial f(z) = \{d | f(z) \geq f(\bar{z}) + d'(z-\bar{z}) \text{ for all } z \in F\} ;$$

when z is a differentiable point of f, $\partial f(z) = \nabla f(z)$. Furthermore, there exists a set of subgradients which are the limits of the one-sided directional derivatives at z. These particular subgradients, referred to as "primitive subgradients," may be used in convex combinations to generate any member of the subdifferential. This presentation is made in greater detail by Rockafellar (1970) and Sandi (1979).

Demjanov (1968) demonstrated that at any nondifferentiable point of a convex function f, there exists a directional derivative $-d_m = f'(z) \in \partial f(z)$ which is the locally best direction of descent. That direction is the minimum norm of the subdifferential; that is,

$$d_m = \text{argmin} \{ ||g||^2 \text{ st } g \in \partial f(z) \}$$

which is the point of the subdifferential closest to the origin. Furthermore, a point z^* is an unconstrained optimum if and only if $-d_m = f'(z^*) = 0$.

Although using the minimum norm may be attractive as a descent direction, the required primitive subgradients are usually not all known; the usual approach is to find a single subgradient at each iteration and update the allocations using it along with a step sequence satisfying (2.8). The method works because any element of the current subdifferential forms an acute angle with the true direction to the optimal point, so by taking a step of the appropriate length, we move pointwise closer to the optimal solution at each iteration. However, it is not necessary for the objective function value to improve at each iteration. The sequence achieves an optimal point when a zero-subgradient is encountered (Held, Wolfe, Crowder, (1974)).

The method is attractive due to its extreme simplicity and is commonly used as a mechanism for multiplier adjustment in Lagrangean relaxations (e.g., Fisher, 1985) and for capacity reallocation in resource-directive algorithms for the MCTP (see Kennington and Shalaby (1977), Rosenthal (1983), or Allen (1985)).

The choice of stepsize sequence is critical to the success of subgradient methods. For instance, the harmonic series, $s^k = 1/k$, satisfies (2.8) but exhibits poor convergence in practice (Bazaraa and Sherali, 1981). Convergence has also been demonstrated for

$$s^k = n(v^* - f(z^k))/||g^k||^2$$

where $0 < n \leq 2$ is a scalar multiplier, v^* is the optimal objective function value, and g^k is the norm of the current subgradient (Polyak (1967)). Since v^* is generally not known, several researchers (for example, Kennington and Shalaby (1977) and Bazaraa and Sherali (1981) have replaced it by approximating values such as upper bounds or combinations of bounds. Goffin (1977) discusses a geometric stepsize progression which exhibits quadratic convergence, but may not achieve an optimal point. Some good experiences have been reported for these methods, but these "heuristic" stepsizes frequently produce poor convergence, or, worse, may converge to non-optimal solutions. It is common practice to include in the process a heuristic rule for modifying the value of n when progress is slow.

We now present the specific application of the subgradient approach to the MCTP, first to the resource direction routine in Section A, then to the Lagrangean routine in Section B, and finally present the overall procedure in Section C.

A. GENERATING UPPER BOUNDS VIA RESOURCE DIRECTION

We use the subgradient approach in resource direction to perform the update, $y^{k+1} = \Pr[y^k + s^k d^k]$ for $y^k, y^{k+1} \in Y$, to solve the problem $\min_y \bar{V}(y)$ st $y \in Y$ where

$$\begin{aligned} \bar{V}(y) = \min \quad & cx \\ \text{st} \quad & Nx = b_2 \\ & x \le y \\ & x \ge 0 \; . \end{aligned}$$

Notice that changes in capacity allocations act as parametric changes to the right-hand side of the subproblem, so $\bar{V}(y)$ is convex with respect to changes in y: a proof of this is given in Kennington and Helgason (1980).

To identify an appropriate subgradient, we first recall (2.7). For any allocation y producing a feasible solution, we have $\bar{V}(y) = u_2 b_2 - u_3 y$, where u_2 and u_3 solve the dual program. The following proof from Kennington and Helgason (1980) shows that $-u_3$ is a subgradient of the function $\bar{V}$ at y:

Lemma 2.1: Let $y \ge 0$ be any feasible allocation to $\bar{V}(y)$ and (u_2, u_3) be the associated optimal dual solution. Then $-u_3$ is a subgradient of $\bar{V}$ at y.

Proof: Let $y, \hat{y} \in Y$ be any feasible allocations, with optimal dual solutions (u_2, u_3), $(\hat{u}_2, \hat{u}_3)$ in $\bar{V}(y)$,

$\bar{V}(\hat{y})$, respectively. Then

$\bar{V}(\hat{y}) - \bar{V}(y) = \hat{u}_2 b_2 - \hat{u}_3 \hat{y} - (u_2 b_2 - u_3 y) \geq$
$u_2 b_2 - u_3 \hat{y} - (u_2 b_2 - u_3 y) = -u_3(\hat{y} - y)$, or
$\bar{V}(\hat{y}) \geq \bar{V}(y) - u_3(\hat{y}-y)$ so $-u_3$ is a subgradient of $\bar{V}$ at y. QED.

Therefore, a subgradient is directly available from the subproblem at each iteration as the negative of the optimal dual u_3, for allocation y.

If $-u_3$ were a feasible direction, the subgradient reallocation would be $y^{k+1} = y^k - s^k u_3{}^k$. However, since $-u_3$ generally yields infeasible allocations, we project the infeasible reallocation $\hat{y} = y^k - s^k u_3{}^k$ to a feasible allocation, y^{k+1} by solving the quadratic program

$$\begin{aligned} &\min \; ||(y^{k+1} - \hat{y})||^2 \\ &\text{st} \;\; y^{k+1} \in Y \end{aligned}$$

An equivalent form of this problem is

$$\min \sum_{p} \sum_{j} (y_{pj}{}^{k+1} - \hat{y}_{pj})^2 \quad \text{st} \quad y^{k+1} \in Y ,$$

which decomposes on j. Solving a quadratic program for each jointly constrained arc at each iteration is a burdensome task, so a heuristic projection is generally used which does not involve the quadratic functions.

Unfortunately, computational results of Allen (1985), for instance, indicate that the method is unable to obtain near-optimal solutions on even moderate-sized problems, even when the bulk of the solution time is spent on the primal reallocations.

Two factors seem to contribute to this failure. First, the nature of the subgradient itself is that no primitive element of the subdifferential is guaranteed to be an improving direction. Second, the space of the primal variables is relatively large, complicating the problem. That is, if $J_T = \{j \mid \sum_p x_{pj}^* = b_{1j}\ x^* \text{ optimal in MCTP}\}$, then in applying the subgradient approach to the Lagrangean problem, we work with $O(|J_T|)$ dual variables at each iteration; in the primal resource allocation, we make $O(|J_T| \cdot |P|)$ reallocations. Thus, as the number of products grows, the master problem becomes substantially harder.

We implement a projection method which ignores $0 \le y_p \le b_1$ and maintains the bounds externally by, in essence, a simple minimum ratio test. The resulting projection can be solved analytically, and applied in practice through a simple set of calculations. Therefore, although it does not relieve the previously mentioned shortcomings of the subgradient method, it does not add to them. Furthermore, it preserves theoretical convergence of the method for a stepsize sequence satisfying (2.8).

If we let y^k and $u_3{}^k$ be the allocation and dual variables obtained at iteration k, the infeasible reallocation $\hat{y}$ and the feasible (projected) allocation y^{k+1}, then we may express the last two allocations as

$$\hat{y} = y^k - s^k u_3{}^k$$

and

$$y^{k+1} = y^k - s^k \hat{u}_3{}^k \ .$$

The resulting quadratic program is

$$\begin{aligned} &\min \ ||y^{k+1}-\hat{y}||^2 \\ &\text{st} \ \ Ay^{k+1} = b_1 \ . \end{aligned}$$

Recognizing that $y^{k+1}-\hat{y} = -s^k\hat{u}_3{}^k + s^k u_3{}^k$, that $Ay^k = b_1$, and that s^k is merely the scale factor, an equivalent problem is

$$\begin{aligned} &\min \ ||u_3{}^k-\hat{u}_3{}^k||^2 \\ &\text{st} \ \ Au_3{}^k = 0 \ . \end{aligned}$$

For a single constraint j, dropping the iteration count, the resulting problem is

$$\begin{aligned} &\min \quad \hat{u}_{3j}'\hat{u}_{3j} - \hat{u}_{3j}'u_{3j} \\ &\text{st} \quad \vec{1}'u_{3j} = 0 \ . \end{aligned}$$

The Kuhn-Tucker conditions for this problem are

$$\hat{u}_{3j} - u_{3j} - \vec{1}\cdot w = 0 \qquad (2.9)$$

where w is the dual variable associated with the constraint. Premultiplying by $\vec{1}'$, recognizing that $\vec{1}'u_{3j} = 0$ and solving we find $w = -(\vec{1}'\vec{1})^{-1}\vec{1}'u_{3j}$. Substituting into (2.7) and rearranging, the familiar projection

$$u_{3j} = (I - \vec{1}(\vec{1}'\vec{1})^{-1}\vec{1}')u_{3j}$$

is obtained. Observing that $(\vec{1}'\vec{1})^{-1} = 1/|P|$ and rewriting to summation form, we see that the projected subgradient for the p-th product on arc j is

$$\hat{u}_{3pj} = [(|P|-1)u_{3pj}/|P|] - \sum_{p'\neq p} u_{3p'j}/|P| = u_{3pj}-\bar{u}_{3j}$$

where

$$\bar{u}_{3j} = \sum_{p} u_{3pj}/|P|$$

The associated reallocation update is $y_p^{k+1} = y_p^k - s^k(u_{3p}-\bar{u}_3)$, where s^k is from an appropriate stepsize sequence, $\{s^k\}$.

Since the bounding constraints were ignored, $y_{pj} < 0$ or $y_{pj} > u_j$ may result. In this situation we perform a minimum ratio test, setting $0 \leq s_j' \leq s_j^k$ such that $0 \leq y_{pj} \leq b_{ij}$ for all p on arc j. We drop all products with $x_{pj} = 0$ and $u_{3pj} < \bar{u}_{3j}$ on arc j, recomputing $\bar{u}_{3j}$ and reallocating among the remaining products to simplify the process computationally. In addition, if some $x_{p'j} = b_j$ with $u_{3p'j} = \min_p(u_{3pj})$, no reallocation is performed for that arc.

Finally, if $u_{3pj} = \bar{u}_{3j}$ for all p on some arc j, then no reallocation occurs on that arc.

We write the procedure as follows, given x^k, $u_3{}^k$ from the current subproblem, and a stepsize s^k:

```
REALLOT:  Compute $\bar{u}_{3j}{}^k = (A_j u_3)/|P|$
          {For each arc, j:
            Cond 1:  If there is a p' with $x_{p'j} = b_{1j}$
                     and $u_{3p'j} = \min_p(u_{3pj})$,

                     let $y_{pj}{}^{k+1} = y_{pj}{}^k$
            Cond 2:  If $u_{3pj} = \bar{u}_{3j}$ for all p,
                     let $y_{pj}{}^{k+1} = y_{pj}{}^k$
            Cond 3:  Identify $P_N = \{p | x_{pj} = 0, u_{3pj} \geq \bar{u}_{3j}\}$
                     If $P_N \neq \emptyset$, recompute
                       $\bar{u}_{3j} = \sum_{p \notin P_N} u_{3pj}/(|P|-|P_N|)$
                     Over all $p \notin P_N$
                        set $y_{pj}{}^{k+1} = y_{pj}{}^k - s_j{}^k(u_{3pj}-\bar{u}_{3j})$
                        such that $0 \leq s_j{}^k \leq s^k$ and
                        $0 \leq y_{pj}{}^{k+1} \leq b_{1j}$ for all $p \in P$
          End for}
```

Subgradient reallocation need only be performed for the set of joint arcs which are currently tight. For all other arcs, if there is some $x_{pj} = y_{pj}$ with favorable reduced costs, but $A_j x \leq b_{1j}$ for that j, we perform a simple reallocation in the manner of Rosenthal (1983), taking slack from any p' with $x_{p'j} < y_{p'j}$ until it is all used. Thus, in the solution process, all available slack on a joint arc j is offered to each product until it is consumed, at which point that arc becomes a candidate for subgradient reallocation.

In practice, we first perform simple reallocations until no further improved solutions are found, and then incorporate subgradient reallocations into the ensuing iterations for tight joint constraints. The set of tight constraints is updated with each pass through the subproblems. We note that if at any iteration k, there is no arc j to which a simple reallocation may be applied and all arcs with $A_j x = b_{1j}$ also have $u_{3pj} = \overline{u}_{3j}$ for all p, we have achieved an optimal point (Rosenthal, 1983).

Unfortunately, the optimality condition is not frequently achieved, nor is a useful lower bound available from the dual information of the restricted problems, so we establish lower bounds on MCTP through a Lagrangean relaxation routine in hopes of terminating through bound convergence. The Lagrangean approach is explained in the following section.

B. LAGRANGEAN RELAXATION USING CONJUGATE SUBGRADIENT DIRECTIONS

Lagrangean relaxation is a frequently-used device to assist in solving linear programs in which a set of complicating constraints are placed into the objective function with penalty multipliers which are estimates of the optimal dual multipliers to the original problem. Recalling Chapter I, the problem becomes

$$\text{LR} \qquad \max_{u_1 \le 0} \ \min_{x \ge 0} \ cx - u_1(Ax-b_1) \quad \text{st} \quad x \in F,$$

which is frequently solved by fixing $u_j \le 0$ and solving

$$\text{LR}(u_1) \qquad \underline{V}(u_1) = \min(c-u_1A)x + u_1b_1 \quad \text{st} \quad x \in F \ ,$$

to yield a lower bound on the original problem. A new u_1 is then found and $\text{LR}(u_1)$ resolved, repeating until optimality is achieved.

A common method of accomplishing the u_1 updates is by subgradient optimization, where $g = (Ax-b_1)$ is a subgradient and

$$u_1^{k+1} = \min(0, u_1^k - s^k g^k)$$

is the update. The "min" operator solves the projection $\min||u_1^{k+1} - \hat{u}_1||^2$ st $u_1^{k+1} \le 0$ (see for instance, Fisher, 1985). However, the method has several drawbacks. First, due to the nature of the subgradient, the bound is not always improved, and due to the sensitivity of the process to choice of s^k, convergence may be slow, or the process may converge to a non-optimal point. Due to the relaxation, it is not necessary that a primal feasible solution ever be generated and, since we only determine one primitive subgradient at each iteration, we are unlikely to find a 0-subgradient and thus recognize an optimal solution.

In this section we introduce a procedure which seeks to overcome some of these difficulties by retaining information on previous subgradient directions in order to approximate the subdifferential at the current point. The concept, developed originally by Wolfe (1975) involves computing a "conjugate subgradient" direction at each iteration which is the nearest point to the origin of a set of m retained subgradients. The method relies on the property that the subdifferential of any point can be arbitrarily well approximated by the convex hull of gradients of the points in its immediate neighborhood (see Rockafellar, 1970, for details). The algorithm generates a sequence of points $\{u_1^k\}$ and subgradients $\{g^k\}$ and computes a search direction d^k as the solution to

$$
\text{(CD)} \qquad \begin{aligned}
\min \quad & ||d||^2 \\
\text{st} \quad & \sum_{n=0}^{m} w_n g^{k-n} = d \\
& \sum_{n=0}^{m} w_n = 1 \\
& w_n \geq 0 \text{ for } n = 1,m
\end{aligned}
$$

for some integer m. At some step of the algorithm, some subsequence of points, $\{u_1^k, u_1^{k-1}, \ldots, u_1^{k-m}\}$ is close enough together that

$$\partial f(z^k) \doteq \operatorname{conv}\{g^k, g^{k-1}, \ldots, g^{k-m}\}$$

and

$$d^k \doteq 0 \ .$$

When this occurs, we are sufficiently close to the optimal solution to terminate with u_1^k as an approximation for u_1^*.

Wolfe's algorithm calls for a single variable search to optimize the objective in the conjugate subgradient direction; this is computationally expensive since the inner minimization involves solving the set of network subproblems for various step lengths. So, we find s^k approximately. Define a nominal stepsize s_n as

$$s_n = n(\overline{V} - \underline{V})/||g^k||$$

where $||g^k||$ is the Euclidean norm of the current subgradient, and n is a scalar multiplier. $\overline{V}$ is the current upper bound, and $\underline{V}$ is the current lower bound. Then we evaluate $W(n) = \underline{V}(u_1^k + s_n d^k)$ for n = 0,1, and 2. Since the objective function value of the dual of a linear program is piecewise-linear and concave as a function of the multiplier values (Bazarra and Shetty (1979)) this surface can be approximated by fitting a quadratic interpolating function through these three observations. Using the Newton-Gregory forward equation to interpolate on equally spaced data, we calculate

$$\hat{W}(\tilde{n}) = W(0) + \tilde{n}W1 + .5\tilde{n}(\tilde{n}-1)W2 \qquad (2.10)$$

for $0 \leq \tilde{n} \leq 2$, where

$$W1 = W(1) - W(0)$$

and

$$W2 = W(2) - 2W(1) - W(0) \ .$$

Details may be found in Gerald and Wheatley (1984), for example. To compute the appropriate step length, we select

$$n^k = \text{argmax } (\hat{W}(\tilde{n}) | 0 \leq \tilde{n} < 2)$$

and set $s^k = n^k(\bar{V}-\underline{V})/||g^k||$. The resulting multiplier update is $u_1^{k+1} = \max(0,u_1^k+s^kd^k)$, and we solve for $\underline{V}(u_1^{k+1})$ to complete the approximate line search.

Solving $LR(u_1^{k+1})$ provides a new subgradient $g^{k+1} = (Ax^{k+1}-b_1)^+$ which is added to the retained set. We are then ready to compute a new conjugate direction using (CD).

Since we are approximating the subdifferential, d^k will not always be an ascent direction. In that case, we simply take a small step in the direction d^k, generate g^{k+1} to improve our local approximation of the subdifferential, compute a new d^{k+1}, and continue.

When d^k is near zero, we must insure that the sequence of u_1's generated is suitably close together to adequately approximate the subdifferential at u_1^k. Wolfe suggests this amounts to the check $\sum_{n=1}^{m} ||u_1^{k-n} - u_1^{k-n+1}|| < M$ for some $M > 0$. If this condition holds, u_1^k is accepted as near optimal; if not, all retained subgradients are dropped and the process is restarted from u_1^k.

It is possible to produce feasible capacity allocations from Lagrangean solutions, which may be used as starting points for resource-directive procedures. These allocations must be integer to preserve integer arithmetic in the network subproblems. Given a current value for x, the capacity allocation y may be computed by

$$\text{(CA)} \qquad y_{pj} = \begin{cases} \Pr[x_{pj} \cdot (b_{1j}/\sum_p x_{pj})] & \text{if } j \in J_V \\ \\ \Pr[x_{pj} + (\sum_p x_{pj} - b_{1j})/|P|] & \text{if } j \notin J_V \end{cases} \qquad (2.11)$$

where $\Pr[\cdot]$ indicates a projection onto the set of integers satisfying $\sum_p y_{pj} = u_j$ and $y_{pj} \geq 0$, integer for all p and $j \in J$. In practice, simple rounding is sufficient.

Note that the conjugate subgradient approach is also applicable to the resource-direction problem, but we chose not to use the conjugate approach due to excessive storage requirements. While for the Lagrangean we must store m subgradient vectors, in the resource-directive problem we must store $m \cdot |P|$ vectors.

C. A RESOURCE DIRECTIVE PROCEDURE WITH LAGRANGEAN LOWER BOUNDING

We now present a procedure combining the resource direction of Section A with the Lagrangean routine presented in Section B for solving the MCTP. Upper bounds and feasible solutions are obtained via the resource direction, while lower bounds are obtained from the Lagrangean routine. Although we present a specific stepwise arrangement, the resource directive and Lagrangean routines are coupled only by the values of the bounds. They may be performed sequentially, iteratively, or in parallel, exchanging bound information when appropriate.

Define the stopping criterion $e > 0$ as the allowable relative error between bounds, $D > 0$ as the acceptance criterion for a near-zero direction, and $U > 0$ as the allowable Euclidean separation of points used to approximate the current subdifferential. Let $\overline{V}$ and $\underline{V}$ be the current bounds and $\overline{x}$ be the incumbent solution. Also let K be the maximum allowable number of iterations.

The Algorithm RDLB follows:

Input: The network $T = \{I,J\}$, and joint capacity vector b_1, and for each product $p = 1,\ldots,|P|$, a cost vector c_p, and a supply/demand vector b_{2p}

Output: Incumbent solution $\overline{x}$, and incumbent value $c\overline{x}$.

step 0 (Initialize): Select $e > 0$, $D > 0$, $U > 0$, $m \geq 1$, $K \geq 1$ integer, set $k = 0$, $u^o = 0$, $J_V = \emptyset$

```
Solve LR(0), compute $J_V = \{j | A_j x^o - b_{1j} > 0\}$,
      Evaluate $\underline{V}(0)$ obtaining $x^o$
      If $J_V \neq 0$, stop with $x^o$ optimal in MCTP
      Else, $\underline{V} = \underline{V}(0)$
        set $d^o = g^o = (Ax^o - b_1)^+$
        Set $y^o = CA(x^o)$
        Solve $RS(y^o)$ with $x_y{}^*$ optimal, set $\overline{V} = \overline{V}(y^o)$,
          $\overline{x} = x_y{}^*$
        If $(\overline{V} - \underline{V})/\overline{V} < e$, exit.

step 1:   Solve Lagrangean
      1a (Line Search):  Compute $s = n(\overline{V}-\underline{V})/||g^k||$, solve
          $W(n)$ for $n = 1,2$
          Set $n^k = \text{argmax}\{\hat{W}(\tilde{n}),\ 0 \leq \tilde{n} \leq 2\}$
          $s^k = n^k(\overline{V}-\underline{V})/||g^k||$
      1b  (Move to new point):
          Set $u_1{}^{k+1} = \max(0, u_1{}^k + s^k d^k)$
          Solve $LR(u_1{}^{k+1})$
          If $\underline{V}(u_1{}^{k+1}) > \underline{V}$, $\underline{V} = \underline{V}(u_1{}^{k+1})$
          if $(\overline{V}-\underline{V})/\overline{V} < e$, exit with incumbent $\overline{x}$
          Else, compute subgradient $g^{k+1}$ where

          $g_j{}^{k+1} = \begin{cases} A_j x^k - b_{1j} & \text{for } j \in J_V \\ 0 & \text{otherwise} \end{cases}$

          set $J_V = J_V \cup \{j | A_j x - b_{1j} > 0\}$
          set $G^{k+1} = \{g^{k+1}, \ldots, g^{k-m+1}\}$
```

```
1c (Compute New Direction):
    Let $d^{k+1} = \text{argmin}\{||d||^2 \text{ st } d \in \text{conv}(G^{k+1})\}$

    If $||d^{k+1}||$ $\begin{cases} > D, \text{ set } k = k+1, \text{ go to step 1.} \\ \leq D \text{ and } \sum_{n=0} ||u_1{}^{k-n} - u_1{}^{k-n+1}|| \leq u, \\ \quad k = 1, \text{ go to step 2} \\ \leq D \text{ and } \sum_{n=0} ||u_1{}^{k-n} - u_1{}^{k-n+1}|| > u, \\ \quad \text{set } d^o = g^{k+1}, \\ \quad u_1{}^o = u_1{}^k,\ k = 0,\ G^o = \emptyset, \\ \quad \text{go to step 1.} \end{cases}$

step 2 (Resource Direction):
    2a. Solve RS(y) for the current $y^k$
        If $\overline{V}(y) < \overline{V}$, $\overline{V} = \overline{V}(y)$, $\overline{x} = x_y{}^*$
        If $(\overline{V}-\underline{V})/\overline{V} < e$, exit with $\overline{x}$
        Else let $J_T = \{j | A_j x_j{}^k - b_{1j} = 0\}$
    2b. (Reallocate)
        For all $j \in J_T$, perform the procedure REALLOT
          to obtain $y_{pj}{}^{k+1} = y_{pj}{}^k - s_j{}^k(u_{3pj}{}^k - \overline{u}_{3j}{}^k)$
        If $y_{pj}{}^{k+1} = y_{pj}{}^k$ for all $j \in J_T$, exit
        If $k > K$, exit
        Else, go to step 21.
```

Two aspects of this algorithm represent additional computational burdens over a standard subgradient approach which must be justified. First, the direction-finding step requires solving a quadratic program. This is a small program of m variables which may be solved efficiently as a linear complementarity problem using the Kuhn-Tucker conditions (Bazaraa and Shetty, 1979). Since the purpose of computing conjugate directions is to reduce the zigzagging common to subgradient algorithms, the potential for fewer iterations justifies the effort.

Second, the line search requires two additional subproblem evaluations at each iteration. This means, in effect, that 3 normal subgradient steps can be taken for each conjugate step taken. It is not clear that this effort will always be justified. An alternate form of the conjugate subgradient algorithm which overcomes this problem first takes a series of subgradient steps, retaining the subgradients, and then periodically performs a conjugate subgradient step with quadratic approximation. This amortizes the cost of the search over several iterations. After each conjugate step, old subgradients are dropped and a new collection begins. Computational results presented in Chapter IV show that the conjugate directions method does generate a direction resulting in a large increase to the lower bound every few iterations, but is rather slow. However, because the entire procedure performs much worse than other algorithms tested, the alternate forms suggested above have not been tested.

III. PRICE DIRECTIVE DECOMPOSITION ALGORITHMS

In this chapter we develop the theory of dual decomposition and introduce a new decomposition algorithm which solves a penalized version of the original linear problem. This is a procedure which uses restricted simplicial decomposition to construct a nonlinear master problem and conveys price information to the subproblems through the gradient of the objective function. With proper choice of penalty parameters, the subproblems quickly produce good Lagrangean lower bounds on the original problem, and improving primal solutions are produced by resource direction using the (infeasible) penalized master problem solutions.

Both algorithms are applicable to general linear programs and their development here is accordingly general.

A. THE DUAL DECOMPOSITION ALGORITHM

The standard approach in describing the dual decomposition algorithm is to put forth the idea of representing the feasible region of a subset of the constraints of an LP as a convex combination of its extreme points. This reduces the number of constraints, but introduces a large number of variables (the extreme points of the dualized constraints), so the problem becomes one of finding the extreme points which, in convex combination, describe the optimal solution to the original problem. Dantzig-Wolfe decomposition combines this extreme point representation with column generation, which is the process of generating extreme points as required.

If we let X be a matrix whose columns are the extreme points of F, then any x in F may be expressed as

$$x = Xw,\ \vec{1}\cdot w = 1,\ w \geq 0\ ,$$

where $\vec{1}\cdot w = 1$, $w \geq 0$ enforce convex combinations of the extreme points.

Substituting this form yields the Dantzig-Wolfe master problem, at the kth iteration

$$\begin{array}{llcr}
\min & cX^k w^k & \text{(duals)} & (3.1) \\
\text{st} & (Ax^k)w^k \leq b_1 & u_1 & (3.2) \\
 & \vec{1}\cdot w^k = 1 & u_o & (3.3) \\
 & w^k \geq 0 & & (3.4)
\end{array}$$

where X^k is a matrix of extreme points collected so far, and (3.3) and (3.4) are the convexity constraints.

Each time the master problem is solved, we check to see if it is an optimal solution by computing the reduced costs for the non-basic extreme points of F. It is not necessary to check them all since the most negative reduced cost is found by solving the problem

$$\begin{aligned} &\min \quad (c-u_1A)x - u_o && (3.5) \\ &\text{st} \qquad\quad x \in F\ , \end{aligned}$$

which produces an extreme point of F. This is the Dantzig-Wolfe subproblem, which for MCTP is a set of single-product network problems which are easy to solve.

If x^k solves the subproblem at the kth iteration and $(c-u_1A)x^k-u_o < 0$, then the reduced cost associated with x^k is favorable, so x^k is added to the collection of extreme points and we return to the master problem. If $(c-u_1A)x^k-u^o \geq 0$, then the reduced cost for x^k is unfavorable. Since, due to the minimization, it has the minimum reduced cost, then all non-basic extreme points of F have unfavorable reduced costs, and the process terminates with the solution to the previous master problem optimal in MCTP.

A second approach may be taken to the dual decomposition algorithm which considers the decomposition as a cutting plane process (Kelley, 1960), resulting in a master problem which is dual to the standard master problem (e.g., Graves and Van Roy, 1979).

The problem to be solved is

$$\text{(P)} \qquad \begin{array}{lll} \min & cx & \text{(dual variables)} \\ \text{st} & Ax \le b_1 & (u_1) \\ & Nx = b_2 & (u_2) \\ & x \ge 0 & \end{array}$$

whose dual is

$$\text{(D)} \qquad \begin{array}{ll} \max & ub = u_1 b_1 + u_2 b_2 \\ & u_1 A + u_2 N \le c \\ & u_1 \le 0,\ u_2 \text{ free }. \end{array}$$

The feasible regions associated with the various constraints are

$F_{P1} = \{x | Ax \le b_1\}$

$F_{P2} = \{x | Nx = b_2\}$

$F_P = F_{P1}$ and F_{P2} and $\{x | x \ge 0\}$

$F_{D1} = \{u | u_1 A + u_2 N \le c\}$

$F_D = F_{D1}$ and $\{u | u_1 \le 0\}$.

The respective values of the primal and dual problems are written $V(P) = cx$ and $V(D) = ub$.

Let $f(u,x) = cx - u_1(Ax-b_1)$, which we recognize as the Lagrangean of (P) with respect to the constraint set A.

Lemma 3.1: If $x \in F_{P2}$, $x \ge 0$, $u \in F_D$, $u_2 \le 0$, then $f(u,x) \ge ub$.

Proof: Write the identity:

$$ub = cx - u_1(Ax-b_1) - u_2(Nx-b_2) - (c-u_1A-u_2N)x.$$

Rearrange and substitute:

$$ub = f(u_1,x) - u_2(Nx-b_2) - (c-u_1A-u_2N)x.$$

But the two right-most terms are ≤ 0, giving the desired result. QED.

This suggests that by fixing u_1, the following subproblem results:

$$(\text{SUB}(u_1)) \qquad \begin{array}{lll} \min & f(u_1,x) = u_1b_1 + (c-u_1A)x & \\ \text{st} & Nx = b_2 & (u_2) \\ & x \geq 0 & \end{array}$$

This is the Lagrangean relaxation of (P) with respect to A, with u_1 fixed. If (SUB(u_1)) is infeasible, so is (P); otherwise we generate u_1 using a master problem in a convergent algorithm.

Let $u^k = (u_1{}^k, u_2{}^k)$ be a composite dual solution at step k of the algorithm. The following lemmas describe its properties:

Lemma 3.2: Let x^k and $u_2{}^k$ be respective optimal primal and dual solutions of (SUB(u_1)). Then $u^kb = f(u_1{}^k,x^k) = V(\text{SUB}(u_1))$, and $u^k \in F_D$.

Proof: Again, use the identity

$ub = cx-u_1(Ax-b_1)-u_2(Nx-b_2)-(c-u_1A-u_2N)x.$

So $u^kb = f(u_1{}^k,x^k)-u_2{}^k(Nx^k-b_2)-(c-u_1{}^kA-u_2{}^kN)x^k.$

Since $u_2{}^k,x^k$ are optimal in $(SUB(u_1))$, by complementary slackness

$u_2{}^k(Nx^k-b_2) = (c-u_1{}^kA-u_2{}^kN)x^k = 0.$

$(SUB(u_1{}^k)$ optimal implies

$(c-u_2{}^kN)-u_1{}^kA \geq 0$, and $u_1{}^k \leq 0$, yielding $u^k \in F_D$.

QED

According to Lemma 3.2, for any $u_1{}^k \leq 0$, a feasible solution $u^k = (u_1{}^k,u_2{}^k) \in F_D$ can be constructed with value $V(SUB(u_1))$. Let $\bar{u}$ be the best solution to (D) currently known. The following lemma establishes the necessary conditions for improving the incumbent solution, $\bar{u}$.

Lemma 3.3: Let $K = \{k|x^k \in F_{P2},x^k \geq 0\}$. A necessary condition for $V(SUB(u^k{}_1)$ to yield a dual solution value better than the incumbent value $\bar{u}b$ is that $f(u,x^k) \geq \bar{u}b+e$, $k \in K$, for some $e > 0$.

Proof: $x^k \in F_{p2}$ is feasible in $(SUB(u_1))$ for any u_1, so $V(SUB(u_1)) \leq f(u,x^k)$. For $V(SUB(u_1)) > \bar{u}b$, u_1 must satisfy $f(u_1,x^k) \geq \bar{u}b+e$ for $k \in K$ and $e > 0$. QED

The existence of a convergent algorithm for finding $u_1{}^*$ is established as follows.

Lemma 3.4: The sequence $(SUB(u_1))$ with arguments $\{u_1^k\}$ is finite if $\{x \in F_{P2}, x \geq 0\}$ is bounded, $V(D)$ is finite, and at any step k, u_1^k satisfies $f(u_1^k, x) \geq \overline{u}b+e$ for $l = 1,\ldots,(k-1)$ for some $e > 0$.

Proof: $SUB(u_1^k)$ yields basic solutions, x^k, which are finite in number. The lemma follows if any repeated basic solution yields termination of the sequence. By lemma 2, $u^k b = f(u_1^k, x^k)$ and $u^k \in F_D$. If $x^k = x^l$ for some $l < k$, then $u^k b = f(u_1^k, x^k) = f(u_1^k, x^l) \geq \overline{u}b+e$ since u_1^k satisfies $f(u_1^k, x\ ) \geq \overline{u}b+e$ for $l = 1 \ldots (k-1)$. Thus u_1^k is a new incumbent, and with $e > 0$ and $V(D)$ finite, there may only be a finite number of such updates. QED

This criterion leads naturally to the master problem:

$$f(u_1, x^l\ ) \equiv cx^l - u_1(Ax^l - b_1) \geq \overline{u}b+e,\ l = 1,\ldots,k$$
$$u_1 \leq 0$$

Notice that the objective function is unspecified: any objective will suffice. Experience shows that the most recent cut works well in practice. That is,

$$(MP(x)) \quad \max \quad f(u_1, x^k)$$
$$\text{st} \quad f(u_1, x^l) \equiv cx^l - u_1(Ax^l - b_1) \geq \overline{u}b+e,$$
$$l = 1,\ldots,(k-1), \quad u_1 \leq 0$$

MP(x) may be unbounded, in which case any $u_l \le 0$ such that $f(u,x^l) \ge \bar{u}b+e$, $l = 1,\ldots,(k-1)$ will suffice. When $V(MP(x)) < \bar{u}b+e$, the process terminates with an e-optimal solution.

According to the following lemma, a feasible primal solution to (P) can be recovered whenever V(MP(x)) is finite.

Lemma 3.5: Let $w_l \ge 0$, $l = 1,\ldots,(k-1)$ be the optimal dual solution to MP(x) at iteration k. If V(MP(x)) is finite, a primal feasible solution to (P) is

$$x^k = [x^k + \sum_{l=1}^{k-1} w_l x^l]/[1 + \sum_{l=1}^{k-1} w_l]$$

Proof: MP(x) is restated in canonical form as

$$\max \; -u_l(Ax^k-b_l) + cx^k$$
$$-u_l(b_l-Ax^l) \le cx^l - [\bar{u}b+e],\; l = 1,\ldots,(k-1) \quad (w_l)$$

By duality, the optimal dual solution, w, satisfies

$$\sum_{l=1}^{k-1} - (b_l-Ax^l)w_l \le b_l-Ax^k, \text{ yielding}$$

$$A[x^k + \sum_{l=1}^{k-1} w_l x^l] \le [1 + \sum_{l=1}^{k-1} w_l]b_l$$

Therefore, $x^k \in F_{P1}$. Also $\bar{x}^k$ is a convex combination of $x^l \in F_{P2}$, so $\bar{x}^k \in F_P$. QED

Lemma 3.6: If $V(MP(x)) \leq \overline{ub}+e$, then $\bar{x}$ and $\bar{u}$ are e-optimal primal and dual solutions to (P).

Proof: $V(MP(x)) = f(u_1{}^{k+1},x^k) = cx^k - u_1{}^{k+1}(Ax^k - b_1)$ (primal)

$= cx^k + \sum [cx^l - (\overline{ub}+e)]w_l$ (dual)

$= [cx^k + (\overline{ub}+e)][1 + \sum w_l] + (\overline{ub}+e)]$

Rearranging yields

$cx^k - (\overline{ub}+e) = [f(u_1{}^{k+1},x^k) - (\overline{ub}+ e)]/[1 + \sum w_l]$ and $cx^k \leq f(u_1{}^{k+1},x^k) \equiv V(MP(x))$. If $V(MP(x)) \leq \overline{ub}+e$, then $cx^k < \overline{ub}+e$. If u^* is the optimal solution of (D), since $x^k \in f_p$, and

$\overline{ub} \leq u^*b$, $u^*b \leq cx^k \leq \overline{ub}+e \leq u^*b+e$. QED

Finally, we show that once the master problem becomes bounded, it remains that way.

Lemma 3.7: The value V(MP(x)) remains bounded once F(MP(x)) becomes bounded.

Proof: V(MP(x)) is finite if F(MP(x)) is bounded. Once F(MP(x)) becomes bounded, subsequent iterations add constraints and make existing constraints tighter by updating $\overline{ub}$, further restricting the problem. QED

The dual decomposition algorithm, DDC, based on the preceding theory follows.

```
Algorithm DDC:

step 0:  Specify $u_1' \le 0$, $e > 0$, $k = 1$, $\overline{u}b = -\infty$

step 1:  Solve $(SUB(u_1^k))$ for $x^k$, $u_2^k$ (i.e., $f(u_1^k,x^k) \ge \overline{u}b+e, x^k \in F_{p2}, x^k \ge 0$)
         If infeasible, stop with (P) infeasible
         If $u^k b \equiv (u_1^k,u_2^k)b \ge \overline{u}b$, update incumbent solution.

step 2:  Solve $(MP(x))$ for $u_1^{k+1}$
         If V(MP(x))
             $< \overline{u}b+e$, declare $x^k$, u e-optimal for (P), stop
             $\ge \overline{u}b+e$ and finite, $x^k \in Fp2$, $k = k+1$, go to step 1
             $\to \infty$, use any $u_1^{k+1} \le 0$, $f(u_1^{k+1},x^k) \ge \overline{u}b+e$, $k = k+1$, go to step 1
             is infeasible, STOP
```

Since the master problem yields feasible primal solutions, it provides upper bounds,

$$V(MP(x)) \ge cx^k \ge u^*b \ .$$

Retaining all cuts may become too expensive in time or space in solving MP(x). It is possible to conduct the algorithm as a heuristic by retaining only a fixed number of cuts. This may degrade the quality of the solution obtainable at any iteration but does not prohibit convergence. A discussion of cut-dropping strategy is provided in the chapter on computational experience.

The value of e need not be fixed. A large value may be used initially which may be reduced as inconsisten-cies are encountered in MP(x). Parametric analysis may be used to find a maximum e, or e may be reduced in a fixed algorithmic sequence of relaxations of cut aspirations.

As a practical matter, the sequence of master problems behaves much like a first-order descent method in convex nonlinear programming. Each cut is a tangential approximation to the objective function as can be seen in Figure 3.1. Just as in nonlinear programming, we can expect oscillation as the solution sequence progresses.

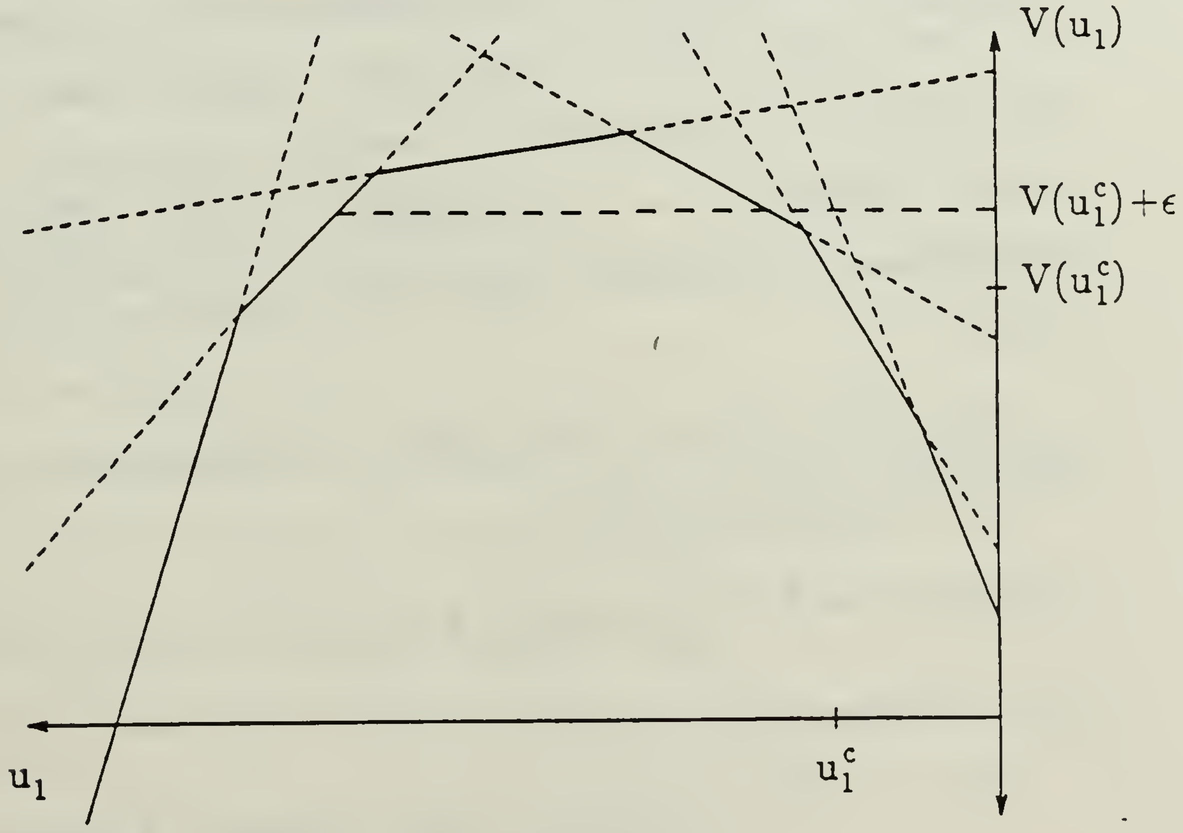

Figure 3.1 Tangential Approximation to Lagrangean Dual Function

In order to deal with such oscillation, a local neighborhood (or trust region) may be specified for u_1. Initially, this neighborhood is relaxed to encompass values of u_1 sure to contain the optimum (i.e., viewing u_1 as a penalty per unit of violation of joint capacity constraints, we wish to assure a feasible completion). Subsequently, the trust region can be restricted when oscillation is apparent (e.g., whenever V(MP(x)) is non-monotonic improving).

As an additional stabilizing influence, we introduce decomposition goals for the master problem variables. These goals may be violated at a small linear penalty cost.

All the essentials for convergence are preserved as long as the cuts are satisfied hierarchically before the goals. In the same vein, the trust region and cut dropping heuristics do not present a serious impediment in practice. Brown, Graves, and Honczarenko (1983) developed similar mechanisms for primal goal decomposition of mixed integer models.

DDC cuts can be restricted to Dantzig-Wolfe cuts if we force a maximal solution:

DDC cuts $\quad cx^l - u_1(Ax^l - b_1) \geq (\bar{u}b + e), \qquad l = 1, \ldots, K_j$

DDW cuts $\quad \max\ u_o$

st $\quad cx^l - u_1(Ax^l - b_1) \geq u_o + u_1 b_1, \quad l = 1, \ldots, K$.

In this sense the dual of the Dantzig-Wolfe master problem is equivalent to the DDW master problem. Taking the dual of (3.1)-(3.4) yields

$$\begin{aligned} \max \quad & u_1 b_1 + u_o \\ \text{st} \quad & u_1(AX^k) + u_o \leq (cX^k)' \\ & u_1 \leq 0 \ . \end{aligned}$$

Rearranging and writing each constraint individually, the result is $u_o \leq cx^l - u_1(Ax^l)$ for $l = 1,k$. Adding $u_1 b_1$ to each side results in

$$u_o + u_1 b_1 \leq cx^l - u_1(Ax^l - b_1) \text{ for } l = 1,\ldots,k \ .$$

Before updating DDC to include all the above innovations, we introduce the following notation:

X is the primal incumbent,

X^k is the matrix of extreme points generated up to iteration k,

$\overline{V}$ is the upper bound,

$e, e_f > 0$ are initial and final convergence tolerances,

$R, R_f > 0$ are initial and final trust regions, and

$n_{hold}, n_f \geq 1$, integer are number of cuts held, and total number of cuts allowed, and

w_e is exponential moving average of cut weights.

The updated algorithm DDC1 becomes:

```
step 0: (initialize)
    Specify $e > 0$, $e_f > 0$, $R > 0$, $R_f > 0$, $n_{hold} \geq 1$
    Set $\overline{ub} = -\infty$

step 1: (solve aggregated problem)
        Solve  $\min c_1 x_1$  st  $N_1 x_1 = b_a$, $0 \leq x_1 \leq b_1$
                    where  $b_a = \sum_p b_{2p}$
            to find $u_1^o \leq 0$
        Set initial decomposition goals.

step 2: Solve (SUB $(u_1^k)$) for $x^k$, $u_2^k$
        Generate cut
        If $u^k b \geq \overline{ub}$, update incumbent solution.

step 3: (optimal capacity allocation)
        Set $y^k = CA(x^k)$, solve $RS(y^k)$ with $x_y^k$ optimal
        If $\overline{V}(y^k) < \overline{V}$, update $\overline{V}$, $\overline{x}$
        Generate cut.

step 4: Solve $(MP(x))$ for $u_1^{k+1}$
        Update decomposition goals
        If $V(MP(x)) < \overline{ub}+e$, set $e = \max(e/2, e_f)$

        If $e > e_f$, repeat step 4
        If $V(MP(x))$ not monotonic, set $R = \max(R/2, R_f)$
```

```
step 5:  If V(MP(x))  { ≥ ub+e and finite, $\bar{x}^k \in F$
                      { $\to\infty$, use $u_1^{k+1} \le 0$, $f(u_1^{k+1}, x^2) \ge ub+e$
                      { is infeasible, STOP
         Set cut weights $\bar{w}^{k+1} = .8\bar{w}^k + .2w^k$
         If solving relaxed master problem, recover primal
              solution (i.e., $\hat{x}_p^k = X^k w^k$)
         If $cx_p^k < \bar{V}$, update $\bar{V}$, $\bar{x}$.

step 6: (termination tests)  k = k+1
         If $n < n_f$ and $\overline{ub} < \bar{V}-e$ go to step 2.
```

Cut generation is performed as follows:

```
Generate cut
    n = n+1
    if $n > n_{hold}$ then
         locate slack cut with minimum weight and replace
         it; if no cut is slack, locate taut cut with
         minimum weight and replace it, also relax upper
         bound from recoverable primal solutions.
    Generate cut of desired class (e.g., Dantzig-Wolfe,
         DDC, ...).
```

B. A PENALTY ALGORITHM FOR LINEAR PROGRAMS USING RESTRICTED SIMPLICIAL DECOMPOSITION

Penalty functions have not been seriously considered as vehicles for large-scale mathematical programming in the past because they introduce nonlinearity (Geoffrion, 1970). However, the recent advent of interior point or logarithmic barrier function methods has shown that non-linear approaches to linear programming are at least viable and offer attractive convergence rates and complexity properties compared with simplex-based methods (Karmarkar, 1984; Gill et al. 1986). In this section we develop the theory and present an algorithm based on penalty function concepts which is well-suited to large-scale optimization applications. The method has strong parallels to Dantzig-Wolfe decomposition, using restricted simplicial decomposition to produce a nonlinear master problem and linear subproblems. The process incorporates Lagrangean lower bounds in a natural way, but produces infeasible solutions in the master problem. We show how to use capacity allocation/restrictions on these infeasible solutions to produce improving upper bounds. The result is a convergent algorithm which displays a superior convergence rate in practice. To set the stage, we begin with a discussion of penalty function theory.

Penalty algorithms approximate constrained optimization problems by unconstrained (or partially constrained) problems. This is done by placing the constraints into the objective function with a penalty parameter which exacts a large price for any violation of the constraints. This relaxed approximation to the original problem becomes more accurate as the penalty parameter is increased. Thus, the penalty algorithm generates a sequence of infeasible points which converges in the limit to an optimal solution of the original problem.

From the general form found in (1.6), we introduce the specific penalty form of a linear program,

$$\text{(PP)} \qquad \min\ q(h,x) = cx + P(h,x)$$
$$\text{st} \quad x \in F$$

where $P(h,x) = \frac{h}{t}||Q(x)||_t^t$, $||\cdot||_t$ indicates the tth norm, $Q(x) = (Ax-b_1)^+$, $(Ax-b_1)^+ = \max(0,Ax-b_1)$, scalar $h > 0$, $1 \leq t \leq 2$, and $F = \{x|Nx = b_2, 0 \leq x_p \leq b_1\}$. This is a common form found in any nonlinear programming text (see, for instance, Bazaraa and Shetty, 1979, or Luenberger, 1984). Recalling that J_v is the set of currently violated constraints, the objective function may also be written as

$$\min\ q(h,x) = cx + \sum_{j \in J_v} \frac{h}{t} Q_j(x)^t \ . \tag{3.2}$$

Since the penalty terms are convex for $h > 0$, the objective function remains convex, and for $t \neq 1$, the objective function is differentiable everywhere. Convexity is proved for the quadratic penalty function in Appendix

A: convexity arguments for other forms may be found in Bertsekas (1982b) and Luenberger (1984), for example.

Due to the convexity of $q(h,x)$, we may employ the standard mathematical result that $\bar{x}$ is optimal in $q(h,x)$ if and only if first-order stationary conditions are met:

$$\nabla q(h,\bar{x})(x-\bar{x}) \geq 0 \qquad (3.3)$$

for all $x \in F$. For $\nabla q(h,\bar{x})(x-\bar{x}) < 0$, $(x-\bar{x})$ represents an improving direction.

The following penalty function characteristics, stated in terms of PP, are taken from Luenberger (1984). Let $\{h^k\}$ and $\{x^k\}$ be sequences of penalty parameters and associated optimal solutions, with $h^{k+1} \geq h^k$, and $h^o > 0$. Then

$$\begin{aligned} q(h^k,x^k) &\leq q(h^{k+1},x^{k+1}) \\ P(x^k) &\geq P(x^{k+1}) \\ cx^k &\leq cx^{k+1} \end{aligned}$$

and

$$cx^* \geq q(h^k,x^k) \geq cx^k ,$$

where x^* is optimal in MCTP. Thus, solving the penalty problem for any $h = h^k \geq 0$ produces a lower bound on MCTP and

$$\lim_{k\to\infty} q(h^k,x^k) = cx^* \text{ with } x^k \in x^* .$$

This large-scale nonlinear version of the original linear problem is useless without an effective means of solution. We base our solution algorithm on the promising method of restricted simplicial decomposition.

Restricted simplicial decomposition (RSD) was developed by Hearn et al. (1984) to solve large-scale pseudo-convex nonlinear optimization problems. The method decomposes the original problem into a nonlinear master problem and a set of linear subproblems (assuming linear constraints). At each iteration, the subproblems generate a new extreme point of the feasible region for the master problem, and the master problem minimizes the original objective function over a simplex of retained extreme points. The master problem in turn provides new cost information to the subproblems via $\nabla f(\hat{x})$ where f is the objective function of the master problem and $\hat{x}$ is the current solution. The process terminates when no favorable extreme points remain. It is termed "restricted" since only a fixed number r of extreme points are retained at each iteration.

When r = 1, RSD specializes to the algorithm of Frank and Wolfe, 1956. In this case the master problem becomes a minimization on the line joining the current point x and the extreme point just generated. Many researchers have noted the slow convergence of the Frank-Wolfe algorithm (Ali, et al., 1978; Meyer, 1974; Wolfe, 1970); a simple example taken from Wolfe (1970) makes this evident. Suppose we are trying to find the point of a polytope which is closest to the origin: in Figure 3.1, this is the midpoint of the base of the triangle. At each iteration, linearizing the objective function and solving the resulting LP leads to vertex B or C of the triangle. Thus, as the algorithm progresses, the line search proceeds along a direction nearly orthogonal to the gradient $\nabla f(x^k)$ causing zigzagging. However, simply by retaining two extreme points and optimizing over the simplex formed by the extreme points and the current iterate, the problem in Figure 3.2 is solved optimally on the second iteration. Indeed, Hearn's computational results show significant improvement as r is increased from 1 (Hearn, et al., 1984).

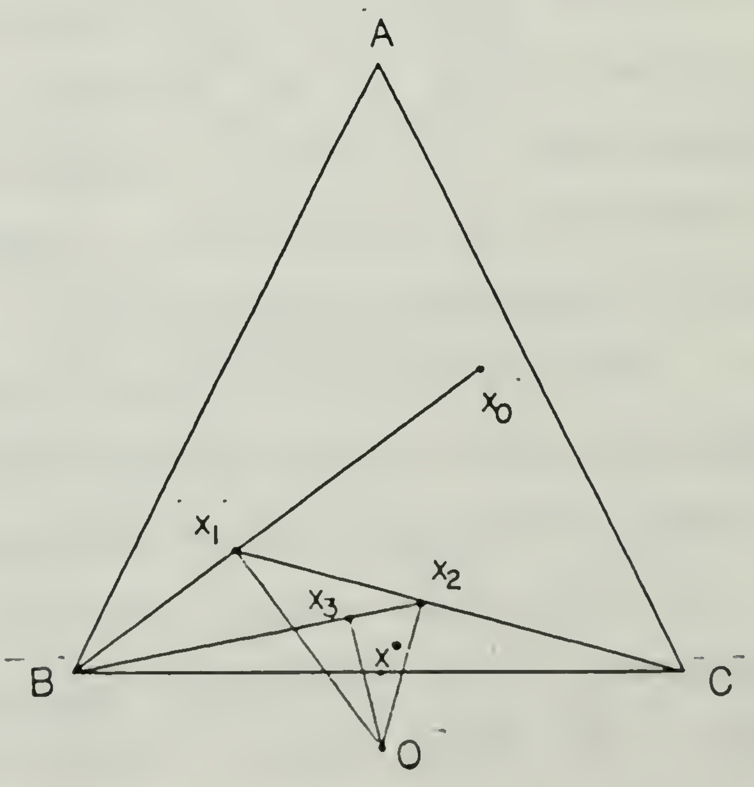

Figure 3.2 Frank-Wolfe Zigzagging Example

Using this method on PP decomposes the problem into a master problem, which minimizes q(h,x) over the set a retained extreme points, and a subproblem consisting of a set of decoupled network flow problems with modified prices. The master problem is simple to solve since it has only a convexity constraint like (3.3) and nonnegativity constraints on a number of variables only equal to the number of retained extreme points. Furthermore, Hearn shows that the method is convergent even if the master problem is only solved approximately.

The decomposition of the penalized MCTP is formed in the following manner. Let X be a matrix whose columns are retained extreme points of F, and r be the number of points retained. Then the master problem becomes

$$\text{(MP)} \qquad \min\ q(h^k, X^k w) = cX^k w + P(h^k, X^k w)$$

$$\text{st:} \qquad \vec{1}\cdot w = 1$$

$$w \geq 0\ .$$

This resembles the Dantzig-Wolfe master problem, except that the penalized joint capacity constraints now appear in the objective function. Let $\hat{x}^k = \sum_{n=1}^{r} X_n{}^k w_n = X^k w$ be the optimal solution to MP at iteration k, and J_V be the set of constraints violated at x^k. The corresponding subproblem is

$$\text{(SP)} \qquad \min\ q(h^k, \hat{x}^k)x \tag{3.4}$$

$$\text{st} \qquad x \in F \tag{3.5}$$

where

$$\nabla q(h^k,\hat{x}^k) = c+\nabla P(h^k,\hat{x}^k) = c+h^k Q(\hat{x}^k)\nabla Q(\hat{x}^k) = c+h^k(A\hat{x}^k-b)^+A \ .$$

Comparing this form to the objective of the Dantzig-Wolfe subproblem (3.5) reveals that $-h^k(Ax^k-b_1)^+$ is an estimate for the dual multipliers, u_1. Thus, we make a new estimate of the optimal multipliers based on the violations in the current master problem. To avoid notational confusion, we let $\hat{u}_1{}^k = h^k(A\hat{x}^k-b^1)^+$ in the rest of this section. Now, if x^k solves SP and

$$\nabla q(h^k,\hat{x}^k)(x^k-\hat{x}^k) \ \geq \ 0 \ ,$$

then $\hat{x}^k$ solves PP for $h = h^k$; otherwise $(x^k-\hat{x}^k)$ is a favorable direction, so we add x^k to the retained extreme points and return to the master problem.

If $\hat{x}^k$ is optimal in PP for $h = h^k$ and x^* is optimal in MCTP, then

$q(h^k,\hat{x}^k) \ \leq \ cx^*$, providing a lower bound on MCTP.

However, since $\hat{x}^k$ is probably not an extreme point of F, we must identify the facet of F containing $\hat{x}^k$ and solve MP exactly to establish this bound. We use other information to provide intermediate bounds.

Recall that the Lagrangean relaxation of the linear program, with u_1 fixed, may be written

$$\text{LR}(u_1) \qquad \min\ (c - u_1 A)x + u_1 b_1$$
$$\text{st} \quad x \in F$$

Using the set J_V, associated with any solution to the master problem, we let

$$\hat{u}_{1j} = \begin{cases} 0 & \text{if } j \notin J_V \\ h(A_j\hat{x}-b_{1j})^{t-1} & \text{if } j \in J_V , \end{cases} \tag{3.9}$$

and rewrite SP as

$$\min \quad (c + \hat{u}_1 A)x$$
$$\text{st} \quad x \in F .$$

Thus, we observe that

$$\underline{V}(\hat{u}_1) = \nabla q(h^k,\hat{x}^k)x^k - \hat{u}_1 b_1 , \tag{3.10}$$

so at each iteration we compute $\hat{u}_1 b_1$ as we compute new subproblem costs and adjust the global lower bound whenever (3.10) exceeds the current lower bound.

Since the penalized problem is convex, it is also possible to linearize the objective function at $\hat{x}^k$ to establish lower bounds via

$$q(h^k,\hat{x}^k) + \nabla q(h^k,\hat{x}^k)(x-\hat{x}^k) \le cx^* .$$

However, the following lemma establishes that the linearization is always dominated by the value of the Lagrangean relaxation with $\hat{u}_1{}^k = h((A\hat{x}^k-b_1)^+)^{t-1}$. For ease of presentation here, we use the non-standard vector form $[(Ax^k-b_1)^+]^t$, meaning $[\max(0,A_jx^k-b_{1j})]^t$ for each element in the vector, and we drop k from x^k and $u_1{}^k$.

Lemma 3.8: Let $\hat{u}_1 = h\{(A\hat{x}-b_1)^+]^{t-1}$ at point $\hat{x} \in F$. Then $q(h,x) + \nabla q(h,\hat{x})(x^k-\hat{x}) \leq \underline{V}(\hat{u}_1) \leq cx^*$, where $x^k = \text{argmin}\ \{\nabla q(h,\hat{x}^k)x \text{ st } x \in F\}$

Proof: For $\hat{u}_1 = h\{(A\hat{x}-b_1)^+]^{t-1}$, $\nabla q(h,\hat{x}) = (c+\hat{u}_1A)$. Now $q(h,\hat{x}) + \nabla q(h,\hat{x})(x^k-\hat{x}) = c\hat{x} + P(h,\hat{x}) + cx^k + \nabla P(h,\hat{x})x^k - c\hat{x} - P(h,\hat{x})\hat{x} = cx^k + P(h,\hat{x}) + \nabla P(h,\hat{x})(x^k-\hat{x}) = (c+u_1A)x^k + P(h,\hat{x}) - \nabla P(h,\hat{x})\hat{x}$ since $\nabla P(h,\hat{x}) = h[(A\hat{x}-b_1)^+]^{t-1}A = \hat{u}_1A$. By definition, $\underline{V}(\hat{u}^1) = (c+\hat{u}_1A)x^k - \hat{u}_1b_1 \leq cx^*$, so our result is proved if $\nabla P(h,\hat{x})\hat{x} - P(h,\hat{x}) \geq \hat{u}_1b_1$. Now $\nabla P(h,\hat{x})\hat{x} - P(h,\hat{x}) = \hat{u}_1Ax - \frac{1}{t}h([(A\hat{x}-b)^+]^{t-1})'(A\hat{x}-b)^+ = \hat{u}_1A\hat{x} - \frac{1}{t}\hat{u}_1(A\hat{x}-b_1)^+$. But, since $\hat{u}_{1j} = 0$ whenever $A_j\hat{x}-b_{1j} \leq 0$, we see that

$$\hat{u}_1A\hat{x} - \frac{1}{t}\hat{u}_1(A\hat{x}-b_1)^+ = n(\frac{t-1}{t})\hat{u}_1A\hat{x} + \frac{1}{t}\hat{u}_1b_1 \ .$$

Since $\hat{u}_1A_j\hat{x} \geq \hat{u}_{1j}b_j$ for all j, then $(\frac{t-1}{t})\hat{u}_1A\hat{x} + \hat{u}_1b_1 \geq \hat{u}_1b_1$. QED

The (relaxed) penalty form of MCTP produces a sequence of infeasible x's which is only guaranteed to converge to x^* in the limit. Therefore, the penalty algorithm does not naturally produce intermediate feasible solutions or upper bounds. In general, this is unacceptable, so we rely on the improving nature of the sequence of x's, using projection and restriction to generate intermediate feasible solutions and upper bounds for MCTP.

Suppose $\bar{h}$ is fixed and $\bar{x}$ solves min $q(\bar{h},x)$ for $x \in F$. Then, if $\hat{x}^k$ solves the master problem at iteration k, either $\hat{x}^k = \bar{x}$ or a favorable extreme point is added to the master problem with

$$q(\bar{h},\hat{x}^{k+1}) \leq q(\bar{h},\hat{x}^k) .$$

If $\hat{x}^k = \bar{x}$, increasing $\bar{h}$ to $h' > \bar{h}$ and resolving MP over the same extreme points yields

$$q(\bar{h},\hat{x}^k) \leq q(h',\hat{x}^{k+1}) ,$$

By sampling the sequence periodically as $\hat{x}^k \rightarrow x^*$ and forming a capacity allocation according to (2.11), $y^k = CA(\hat{x}^k)$ (with scaling to allow non-integer solutions), we will form allocations in RS(y) which improve as x improves, thus obtaining improving feasible solutions and, therefore, upper bounds. These resource allocations may be performed whenever desired; we choose to do an allocation every rth iteration where r is the number of retained extreme points.

The following lemma shows that the allocations must eventually converge to an optimal allocation.

Lemma 3.9. Let $\{x^k\}$ be a sequence solving MP as $\{h^k\} \rightarrow \infty$, $y^k = CA(x^k)$, and $x_y{}^k$ solve $RS(y^k)$. Assuming the solutions are scaled to provide integer allocations, as $h^k \to \infty$, $\overline{V}(y^k) \to V^*$ and $x_y{}^k \to x^*$.

Proof: $\lim x^k = x^*$ for the penalty problem, where cx^* is optimal in MCTP (Luenberger, 1984, Chapter 12). Since $y^k = CA(x^k)$, as $h \to \infty$, $y^* = \lim y^k \to CA(x^*)$. Since x^* is feasible, $y_{pj}{}^* \geq x_{pj}{}^*$ for all p and j. Thus $\overline{V}(y^*) = cx_y{}^* \leq cx^*$. But $x_y{}^*$ is feasible in MCTP by construction, so $cx_y{}^* \geq cx^*$. Therefore $cx_{y^*} = cx^*$, and $x_y{}^*$ solves MCTP. QED

The algorithm proceeds by iteratively solving the subproblems and master problem, and periodically solving the restriction to generate new feasible solutions. The value of h is increased whenever

1) $\nabla q(h,\hat{x}^k)(x^k-\hat{x}^k) \geq 0$,
2) RS(y) will be solved in the current iteration, or
3) $\min q(h,\hat{x}^k) \leq \underline{V}$.

The algorithm terminates when a feasible solution is generated in the master problem, since it must be optimal, or when

$$(\overline{V} - \underline{V})/\overline{V} < e$$

for some $e > 0$.

The algorithm uses the following additional notation:

X^k	=	matrix of retained extreme points at iteration
$\hat{x}^k$	=	current master problem solution
x_M	=	previous MP solution considered in current MP
$\hat{X}^k$	=	a matrix formed by augmenting X^k with x_M, written $X^k \cup x_M$
conv(X)	=	convex hull of X
J_V	=	set of capacities violated by the k^{th} solution to MP
a	>	1.0, a scalar multiplier for increasing h
$\bar{x}$	=	current incumbent for MCTP
t	=	power of the penalty function, $1 \leq t \leq 2$
r	$\geq$	1, integer; maximum number of retained extreme points
e	>	0, a stopping parameter
CA(x)	=	a capacity allocation based on x using equation (2.11)
$\bar{V}, \underline{V}$	=	current upper and lower bounds on MCTP.

The Algorithm RSD(P):

Input: The network $T = \{I,J\}$, and joint capacity vector, b_1 and, for products $p = 1,\ldots,|P|$, a cost vector, c_p, and supply/demand vector, b_2

Output: Incumbent solution $\bar{x}$, and incumbent value $c\bar{x}$.

```
step 0 (Initialize):  Select $h^o > 0$, $e > 0$, $a > 1$, $r \geq 1$,
    set $k = 0$
    Solve LR(0), with $\hat{x}^o$ optimal; set $\underline{V} = \underline{V}(0)$
    Form $J_V{}^o = \{j | A_j\hat{x}^o - b_{1j} > 0\}$
    If $J_V{}^o = \emptyset$, stop with $\hat{x}^o$ optimal
    Else, set $y = CA(\hat{x}^o)$
    Solve RS(y), with $x_y{}^o$ optimal; set $\bar{V} = \bar{V}(y)$.
    If $(\bar{V} - \underline{V})/\bar{V} < e$ stop with incumbent $\bar{x} = x_y{}^o$
    Else, set $X^o = \emptyset$, $x_M = x^o$

step 1 (Solve Subproblem)
    Let $x^k = \mathrm{argmin}\{\nabla q(h^k,\hat{x}^k)x \mid x \in F\}$

        where

        $\nabla q(h^k,\hat{x}^k) = (c+\hat{u}_1 A)$ for $\hat{u}_1 = h((A\hat{x}^k-b_1)^+)^{t-1}$
    If $q(h^k,x^k)(x^k-x^k) \geq 0$, $x^k$ solves MCTPP for $h = h^k$
        set $\underline{V} = q(h^k,x^k)$
        If $(\bar{V} - \underline{V})/\bar{V} < e$, exit with incumbent $\bar{x}$
        Else $h^k = a \cdot h^k$
        Go to step 2.
    Else compute $\underline{V}(\hat{u}_1) = \nabla q(h^k,\hat{x}^k)x^k - \hat{u}_1 b_1$
        If $\underline{V}(\hat{u}_1) > \underline{V}$, $\underline{V} = \underline{V}(\hat{u}_1)$
        If $(\bar{V} - \underline{V})/\bar{V} < e$, exit with incumbent $\bar{x}$
        Else
          i)  If $|X^k| < r$, $X^{k+1} = X^k \cup x^k$
         ii)  If $|X^k| = r$, drop the column of $X^k$ which had
              the smallest $w_n$ in convex combination forming
              $x^k$ and replace it with $x^k$ to form $X^{k+1}$

                        $x_M = x^k$
              Set $\hat{X}^{k+1} = X^{k+1} \cup x_M$

Step 2 (Solve Master Problem)

        $\hat{x}^{k+1} = \mathrm{argmin}\ \{q(h^k,x),\ x \in \mathrm{conv}(\hat{X}^k)\}$
```

```
Discard all columns of $\hat{x}^{k+1}$ with $w_n = 0$
Form the set $J_V^{k+1} = \{j \mid A_j\hat{x}^{k+1} - b_{1j} > 0\}$
If k is an integer multiple of r, do a resource
  allocation:
      Set $y = CA(\hat{x}^{k+1})$
      Solve RS(y), with $x_y^{k+1}$ optimal
      If $\bar{V}(y) < \bar{V}$, set $\bar{V} = \bar{V}(y)$, $\bar{x} = x_y^{k+1}$
      If $(\bar{V} - \underline{V})/\bar{V} < e$, exit with incumbent $\bar{x}$
  Else k = k+1
  Go to step 1.
```

One possible difficulty with the penalty form of the objective function is that only the prices associated with the currently violated set of joint capacity constraints are adjusted at each iteration. There is no intermediate adjustment of multiplier values for constraints previously but not currently violated. Furthermore, convergence to optimal multiplier values for MCTP is guaranteed only in the limit, that is, as $h \to \infty$. These potential problems are overcome by the augmented Lagrangean form of MCTP. As the name suggests, the objective function is formed by adding a penalty term to the standard Lagrangean dual form (Hestenes, 1975). Since the theory of this augmentation is developed in terms of equality constraints, we express the joint capacitation constraints of MCTP as

$$(A_jx-b_{1j}) + m_j^2 = 0 \quad \text{for} \quad j \in J .$$

The augmented Lagrangean problem is then

$$\min_{m,x} L(x,\hat{u}_1,h)$$

$$= cx + \sum_{j \in J} [\hat{u}_{1j}\{(A_j x - b_{1j}) + m_j^2\} + \tfrac{1}{2}h|A_j x - b_{1j} + m_j^2|^2], \quad x \in F$$

with $\hat{u}_1$, $h \geq 0$.

Bertsekas (1982b) demonstrates that an equivalent objective not requiring the variables m is

$$L(x,\hat{u}_1,h) = cx + \sum_{j \in J} \{[\max(0,\hat{u}_{1j}+h(A_j x - b_{1j}))]^2 - \hat{u}_{1j}^2\} .$$

This amounts to a multiplier update

$$\hat{u}_1^{k+1} = \max\{0,\hat{u}^k + h(A_j x - b_{1j})\} \quad \text{for all} \quad j \in J .$$

Using vector notation, the problem is stated as

$$\min \quad L(x,\hat{u}_1,h) = cx + P(x,\hat{u}_1,h)$$

$$\text{st} \quad x \in F$$

$$\text{where} \quad P(x,\hat{u}_1,h) = \frac{1}{2h}(||\hat{u}_1'||^2 - ||\hat{u}_1||^2)$$

and $\quad \hat{u}_1' = [\hat{u}_1 + h(Ax - b_1)]^+ .$

By analogy to the penalty version of the algorithm it is easy to see that the subproblem, $\min \nabla L(\hat{x},\hat{u}_1,h)x$ for $x \in F$ may be restated as $\min(c+\hat{u}_1 A)x$, st. $x \in F$, and a lower bound on MCTP is derived by computing $\min \nabla L(\hat{x}^k,\hat{u}_1,h)x^k - \hat{u}_1 b_1$.

Bertsekas points out certain advantages to solving this augmented Lagrangean form of the problem. First, the method will converge to an optimal solution for a finite value of h. Second, the rate of convergence of this augmented form is superior to the penalty form, which depends on the rate of increase of h, thus substituting convergence rate for ill-conditioning concerns in the master problem.

We conclude this section by demonstrating that RSD(P) is a convergent algorithm. Hearn, Lawphongpanich and Ventura (1985) present a proof that RSD either terminates in a finite number of iterations or generates a sequence $\{\hat{x}^k\}$ for which any subsequential limit is a solution. For the penalized version of the MCTP the proof may be simplified, relying on the simple assumption of a bounded feasible region (i.e., $0 \leq x_p \leq b_1 < \infty$.

In preparation for the main result, we first show that solving the master problem for fixed h produces a sequence of improving solutions (Hearn, et al, 1985).

Lemma 3.10: Let $\hat{x}^k$ solve the restricted master problem at iteration k, with $h = \bar{h}$. If x^k is not optimal in PP, then $q(\bar{h}, \hat{x}^{k+1}) < q(\bar{h}, \hat{x}^k)$.

Proof: At iteration k, $\hat{x}^k$ is the current iterate and x^k solves the subproblem with $\nabla q(\bar{h}, \hat{x}^k)(x^k - \hat{x}^k) < 0$. Thus $x^k \in X^{k+1}$ and $\hat{X}^{k+1} = X^{k+1} \cup \hat{x}^k$. Now let $\hat{x}^{k+1} = \text{argmin}\{q(h,x) \text{ s.t. } x \in \text{conv}(X^{k+1})\}$. Then, $q(\bar{h}, \hat{x}^{k+1}) \leq q(\bar{h}, \hat{x}^k)$ since $\hat{x}^k \in \hat{X}^{k+1}$. Now, if $q(\bar{h}, \hat{x}^{k+1}) = q(\bar{h}, \hat{x}^k)$, then $\hat{x}^k$ also minimizes the master problem at iteration k+1 and $\nabla q(\bar{h},\hat{x}^k)(x-\hat{x}^k) \geq 0$ for all $x \in X^{k+1}$. But since $x^k \in \hat{X}^{k+1}$, this contradicts the assumption that $(x^k-\hat{x}^k)$ is an improving direction. QED

The main convergence result is stated as follows:

Lemma 3.11: Let F be a bounded, non-empty set of feasible single commodity network flows. The RSD(P) algorithm either solves PP for $h = \bar{h}$ in a finite number of iterations or converges to an optimal solution $\bar{x}$ as the limit of an infinite sequence. Furthermore, as h is increased to ∞, $q(h,x) \to cx^*$ and $\{x\} \to x^*$, where x^* is an optimal solution to MCTP.

Proof: Let $\bar{x}$ solve min $q(\bar{h},x)$ st $x \in F$. Since F is non-empty and bounded, choose any starting point $\hat{x}^o \in F$, giving $q(\bar{h},\bar{x}) \leq q(\bar{h},\hat{x}^o) < \infty$. Then at any iteration either $\hat{x}^k = \bar{x}$ or

$$q(\bar{h},\bar{x}) \leq q(\bar{h},\hat{x}^{k+1}) \leq q(\bar{h},\hat{x}^k) \leq q(\bar{h},\hat{x}^o) .$$

Letting $d^k = q(\bar{h},\hat{x}^k) - q(\bar{h},\hat{x}^{k+1})$, then $\sum_{k=0}^{\infty} d^k = q(\bar{h},\hat{x}^o) - q(\bar{h},\bar{x})$, so $\lim_{k\to\infty} d^k = 0$. Since only x varies, $\lim_{k\to\infty} |\hat{x}^k-\hat{x}^{k+1}| = 0$ and, since $\lim_{k\to\infty} q(\bar{h},\hat{x}^k) = q(\bar{h},\bar{x})$, then $\{\hat{x}^k\} \to \bar{x}$.

Since $x \in F$, we set h to $\hat{h} > \bar{h}$ and continue the algorithm. By Lemmas 1 and 2, Section 12.1 of

Luenberger (1984), $q(\bar{h},\bar{x}) \leq q(\hat{h},\hat{x}) \leq cx^*$. Finally by Section 12.1, Theorem 1 of Luenberger (1984) letting $h \to \infty$, we have $\lim q(h,x) = cx^*$ and $\{\grave{x}\} \to x^*$. QED

It is not necessary to solve q(h,x) to optimality for each value of h to obtain convergence since all extreme points of the current simplex are elements of F. Thus, whenever it is desirable, we may set a new $\hat{h} > \bar{h}$ and resolve the master problem with new objective function min $q(\hat{h},x)$ over the same simplex to obtain a new $\hat{x}^k$. The subproblem costs are then based on $\nabla q(\hat{h},\hat{x}^k)$. Appropriate opportunities to increase h have already been discussed.

The preceding proof demonstrates that the algorithm is theoretically convergent without resource directive capacity allocations. Indeed, Bertsekas has shown that, for proper choice of sequence {h}, penalty algorithms may be quadratically convergent (1982b). However, in practice, awaiting convergence for large problems may be impractical. Therefore, we use the resource allocation steps to help obtain near-optimality quickly. As with the resource-directive approach discussed in Chapter II, we always employ simple reallocation to obtain the best available solutions.

IV. COMPUTATIONAL EXPERIENCE

The algorithms presented in Chapters II and III have been coded in FORTRAN and tested on various versions of a large scale MCTP using an IBM 3033AP under both VM and MVS operating systems. For comparison, some four and ten commodity problems have been solved using the X-system of Brown and Graves (1974), which exploits the generalized upper bound structure of the complicating constraints in the MCTP and employs advanced starting solutions of the network constraints. These control problems were solved using an IBM 3081K, which is approximately twice as fast as the 3033AP for the X-system.

Throughout this chapter the following acronyms are used:

DDC	= dual decomposition,
RDLB	= resource direction with Lagrangean lower bounds,
RSD	= restricted simplicial decomposition,
RSD(A)	= RSD of the augmented Lagrangean form, and
RSD(P)	= RSD of the penalty form.

The particular test problems used are described in Section A, issues concerning individual procedures are studied in Section B, and comparisons among algorithms are presented in Section C. Whenever the word "gap" is used to describe the quality of a solution, it means $(\overline{V}-\underline{V})/V^*$, where $\overline{V}$, $\underline{V}$, and V^* are the upper and lower bounds, and optimal solution.

A. TEST PROBLEMS

We demonstrate our methods on a transshipment model for delivery over time of military products from production and storage locations to overseas locations to support theatre operations. The model covers five physical echelons, including production plants, storage depots, ports of embarkation, ports of debarkation, and geographic field locations. Road, rail, sea, and air transportation are modelled, and product demands are time-phased. Capacitation occurs primarily on sea and air links, and as throughput capacities on transfer points, requiring replication of some echelons.

The objective of the model is to minimize deviation from on-time deliveries. This is accomplished through a specified set of backlogging arcs with graduated penalties and a system of penalized "artificial" arcs which direct flow around the network to satisfy "undeliverable" demands and to equilibrate supply and demand. The products all use a common unit of flow and incur a common cost on each arc.

The network is abstracted in Figure 4.1--a more detailed description of the model may be found in Staniec (1984). Computational tests use an underlying network of 3,300 nodes and 10,400 arcs, of which 1,071 are subject to non-redundant joint capacity constraints. A set of basic test problems of between four and ten products is used for detailed inter- and intra-method testing. Results of these tests are reported together in the following sections. Three competitive methods are tested on a final 100 commodity test problem in the last section to illustrate the effectiveness of these methods on a typical, large-scale problem.

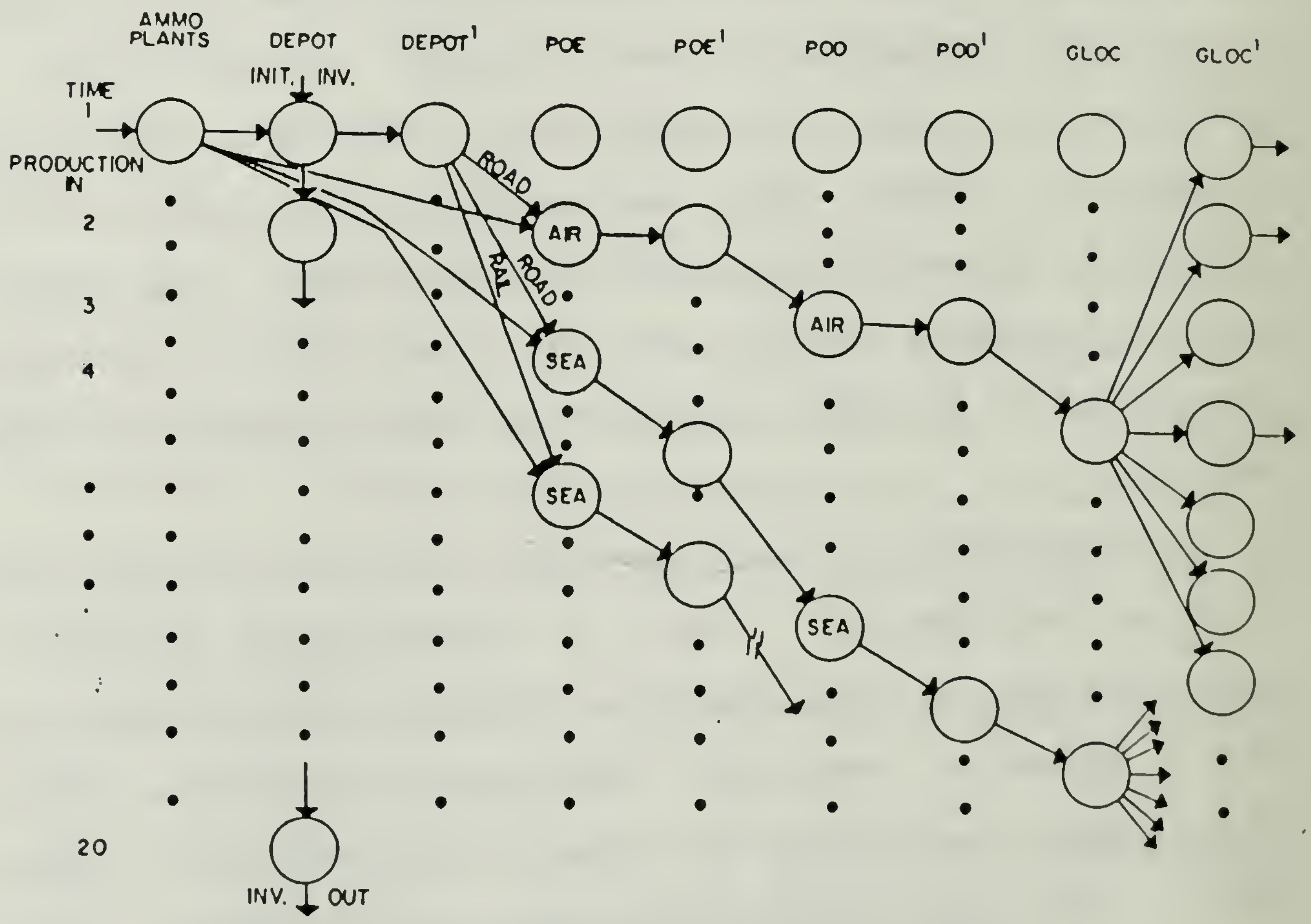

Figure 4.1 Simplified Ammunition Distribution Model

The scope of our computational study is limited to problems drawn from this one generic model, but we think that these problems are generally quite difficult to solve. Because of the explicit system of penalized artificials, the costs range over five orders of magnitude, which have the potential to cause difficulty in real arithmetic. Also, because the model has multiple shipping modes, covers nine spatial echelons, 21 time echelons, and has a large backlogging system, repair of joint capacity violations requires extensive effort. Therefore, these test problems present a rigorous challenge for the solution algorithms.

We have graded the basic test problems used as easy (E) or hard (H) based on two criteria. The first criterion is the number of violations encountered in the first relaxation. This parallels the criterion of Ali, et al. (1984) in evaluating direct simplex methods used on some medium-sized MCTPs: the number of tight joint capacity constraints. The second criterion is the magnitude of the initial violations, which is an indirect measure of the flow changes required to reach the optimal solution. Together these criteria affect the size of the gap between the initial finite upper and lower bounds. This is evident in the RSD and RDLB procedures, where the initial feasible solutions are based upon allocations from the first relaxation: the larger the violations, the poorer the initial allocations. Among the problems we test, the most difficult is 10H, on which RSD and RDLB generate a 56% initial gap.

Test problem specifications and direct solution times by the X-system are presented in Table 4.1. Initial gaps are from the RSD algorithm.

TABLE 4.1

OPTIMAL SOLUTIONS FOR BASIC TEST PROBLEMS

PROD. GRADE	INIT. GAP (%)	OPTIMAL SOL'N	INIT. VIOL.	LARGEST VIOL./ ARC CAP.	X-SYSTEM TIME (SEC*) COLD	HOT
4E	1	340,916,981	3	844/1458	1021	
10E	37	367,135,103	11	4,998/10,500	2586	
4H	14	130,739,585	9	7,185/10,500	461.7	496.5
6H	35	138,525,451	13	10,048/10,500	3015.7	581.0
8H	45	143,526,951	15	13,861/10,500	2431.3	1067.6
10H	54	169,532,339	20	13,861/10,500	5352	1393.5

(*IBM 3081K)

The 4-commodity problems have approximately 13,200 constraints and 41,600 variables: the 10-commodity problems have about 33,000 constraints and 104,000 variables. Hot start solutions use a pure network basis crash, and all these solutions factor the joint capacitation constraints as generalized upper bounds. However, pure network factorization is not employed.

The principal motive for these runs is to establish completely optimal yardsticks for the competing indirect methods.

B. INTRA-ALGORITHM STUDIES

We first discuss the selection of parameter settings for the RSD algorithms, and then we discuss issues involving DDC, including obtaining bounded master problem solutions and cut dropping. For RSD(P), the parameters of interest are the initial value of h, the h multiplier a, and the penalty exponent, t. For RSD(A), we examine h and a, fixing the exponent at 2.0. In our tests, initial values for h are taken as a fraction of the largest cost used in the network, in this case a bypass penalty cost of 62,700. Values tested in RSD(P) were 627.0, 62.7, and 6.27. Other researchers (e.g., Bertsekas, 1982) suggest penalty multipliers in the range of 4 to 10 when solving the problem optimally for each value of h. However, since we increase h frequently based on periodic reallocations and lower bound adjustments, penalty multiplier values between 1.5 and 4 are investigated. Finally, to represent the range of possible exponents we test the algorithm for t = 1.1, 1.5 and 2. The exponent t = 1.0 is not useful in this algorithm since it results in $u_{1j} = h$ for all j in J_V at every iteration.

Table 4.2 summarizes response to changes in the size of the penalty multiplier in RSD(P). The results indicate the upper and lower bounds and final gap based on 21 iterations of the algorithm for problems 4H and 10H. A multiplier of 1.5 seems to work best in the smaller problem, while 4.0

TABLE 4.2

RESPONSE TO CHANGING PENALTY MULTIPLIER

PROD.	MULT. a	INIT h	EXP t	LB (10)	UB (10)	GAP (%)
4H	2.0	6.27	2.0	130.4	130.74	.273
	1.5	6.27	2.0	130.67	130.75	.061
	2.0	6.27	1.5	129.7	130.74	.78
	1.5	6.27	1.5	130.2	130.76	.44
10H	4.0	6.27	2.0	162	170.1	4.79
	2.0	6.27	2.0	163.4	170.6	4.22
	4.0	6.27	1.5	150.2	170.1	11.68
	2.0	6.27	1.5	160.6	170.8	5.99
	4.0	62.7	2.0	156.2	172	9.17
	2.0	62.7	2.0	153.9	171.2	10.09
	4.0	62.7	1.5	163.9	170.6	3.93
	2.0	62.7	1.5	162.1	170.5	4.91

generally works best in the larger problem. However, in both cases, the response is not substantially worse for a multiplier of 2.0. Therefore, we fix the multiplier at 2.0 and concentrate further studies on the other two parameters.

Recalling that in the subproblems the gradient of the penalty term provides an estimate of the optimal Lagrange multipliers,

$$\hat{u}_1^k = h[(A\hat{x}^k - b_1)^+]^{t-1} ,$$

it is evident that h and t work in concert to affect the magnitude of the multiplier estimates at each iteration. We want a combination which produces multiplier estimates large enough to generate new extreme points, yet small enough to allow good lower bound estimates. That is, as $\hat{u}_1{}^k b_1$ grows large, $\underline{V}(\hat{u}_1{}^k)$ degrades as a lower bound estimate. Table 4.3 and Figure 4.2 show the response of RSD(P) to changes in h and t. Results are for 21 iterations of the algorithm, with the penalty multiplier fixed at 2.0 for all cases. In each case we retain a maximum of seven extreme points plus the current master problem solution, and perform a primal resource allocation every seventh iteration.

We summarize the computational results as follows. First, there is a distinct advantage to starting with a small penalty parameter. Initial values of h = (max. arc cost) x 10^{-3} or 10^{-4} provide superior overall performance in the test problems. Second, the penalty exponent $t = 1.1$ shows a definite overall disadvantage to values of 1.5 and 2.0. Both $t = 1.5$ and $t = 2$ perform well for an appropriate matching value of h, but we favor $t = 1.5$ because it provides the best overall final results, apparently reducing the dominance of the most grossly violated arcs when $\hat{u}_1 = h((A_j\hat{x}-b_1)^+)^{.5}$ is computed.

Figure 4.2 graphically displays the interactions between h and t in RSD(P) by depicting upper and lower bounds relative to the optimal solution of problem 10H. The heavy black lines suggest that the best response is with $h = 62.7$, $t = 1.5$, but it appears that any choice of $0 < h \leq 70$ and

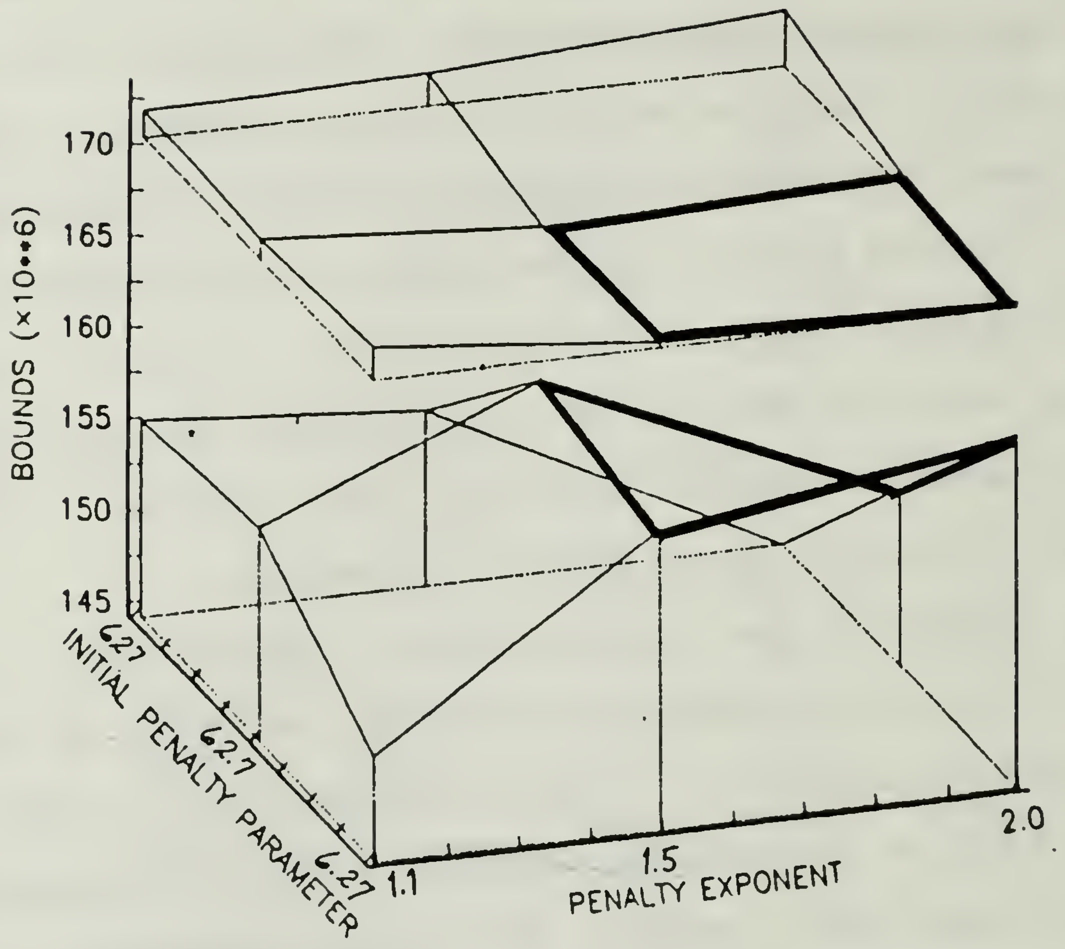

Figure 4.2 Response to Changing Parameters, RSD(P) on Problem 10H

$1.5 \leq t \leq 2$ will perform well. Note that the upper bound is nearly optimal throughout the suggested range.

The parametric responses reported here suggest a simple method for selecting appropriate starting parameters for the penalty algorithm. First, we choose the h multiplier a = 2.0 and set $1.5 \leq t \leq 2.0$. Then we make an estimate, $\hat{u}_{1j}$, of the optimal dual variable associated with the most violated constraint found when LR(0) is solved, and select h so that

TABLE 4.3

RESPONSE TO CHANGES IN PENALTY PARAMETER AND EXPONENT
FOR RSD(P)

Prob.	Init. h	Exponent t = 2.0			t = 1.5			t = 1.1		
		LB (x 10^6)	UB	Gap (%)	LB (x 10^6)	UB	Gap (%)	LB (x 10^6)	UB	Gap (%)
4H	6.27	130.48	130.75	.205	129.9	130.7	.62	120.8	137.0	11.7
	62.7	130.66	130.75	.07	130.55	130.76	.16	125.4	134.4	6.67
	627.0	126.5	131.3	3.59	128.39	130.76	1.8	127.6	133.4	4.36
10H	6.27	163.4	170.6	4.22	160.58	170.8	5.99	150.1	172.3	12.88
	62.7	153.9	171.2	10.09	162.1	170.5	4.91	155.8	171.6	9.22
	627.0	144.3	173.6	16.9	153.8	172.3	10.75	157.8	172	8.25

$$h * \max[(A_j x - b_{1j})^+]^{t-1} \approx \hat{u}_{1j} .$$

Better guesses of $\hat{u}_1$ evidently evoke better performance of the algorithm. In this case, we simply use a fraction of the largest cost as our estimate; more sophisticated estimates may be made by examining differences in costs between least and most expensive paths through the network, or by solving a single problem with aggregated multicommodity supplies and demands. One point is clear from the data: it is better to underestimate the best initial value for h than to overestimate, because the algorithm is quickly self-correcting for low estimates. That is, if the initial penalty is so small that it does not produce a new extreme point, the algorithm immediately (and repeatedly) increases the penalty parameter until it does. Furthermore, when the initial multiplier estimates are good, the subproblems generate extreme points more closely related to an optimal solution. Consequently, both lower bounds and primal solutions quickly benefit. On the other hand, excessively large penalties generate extreme points with little relation to the optimal solution, producing poor lower bounds and degrading the primal solutions obtained through resource allocations.

Figure 4.3 shows the interactions of parameters for RSD(A). In this case, we use a constant penalty exponent t = 2.0, and vary h and a. As indicated by the response surfaces in the figure, the best bounds are obtained for a combination of small initial penalty parameters and smaller multipliers. We fix the multiplier at 1.2 in subsequent tests. Again, the initial value of h may be calculated from a u_1 estimate using

$$h \times [\max(A_j x^o - b_{1j})^+] \approx \hat{u}_{1j} ,$$

and the algorithm quickly corrects for underestimates but recovers slowly from overestimates.

The computational experience presented by Hearn et al. (1984) indicates that performance improves as the number of retained extreme points is increased. In fact, if enough points are retained, theoretically the heuristic becomes a finite algorithm. Performance in our tests improves for up to about eight points, and then levels off. As the number of retained points increases, the time spent in the master problem increases. In the following section, we fix the number of retained points at eight and obtain quite satisfactory performance.

Four performance issues concern us with algorithm DDC.

The first is reaching a bounded condition in the dual master problem (or, equivalently, reaching a feasible solution to the Dantzig-Wolfe master problem). Since the constraint set of the dual master problem may be viewed as forming a piecewise linear tangential approximation of the Lagrangean dual function, we obtain no feasible primal

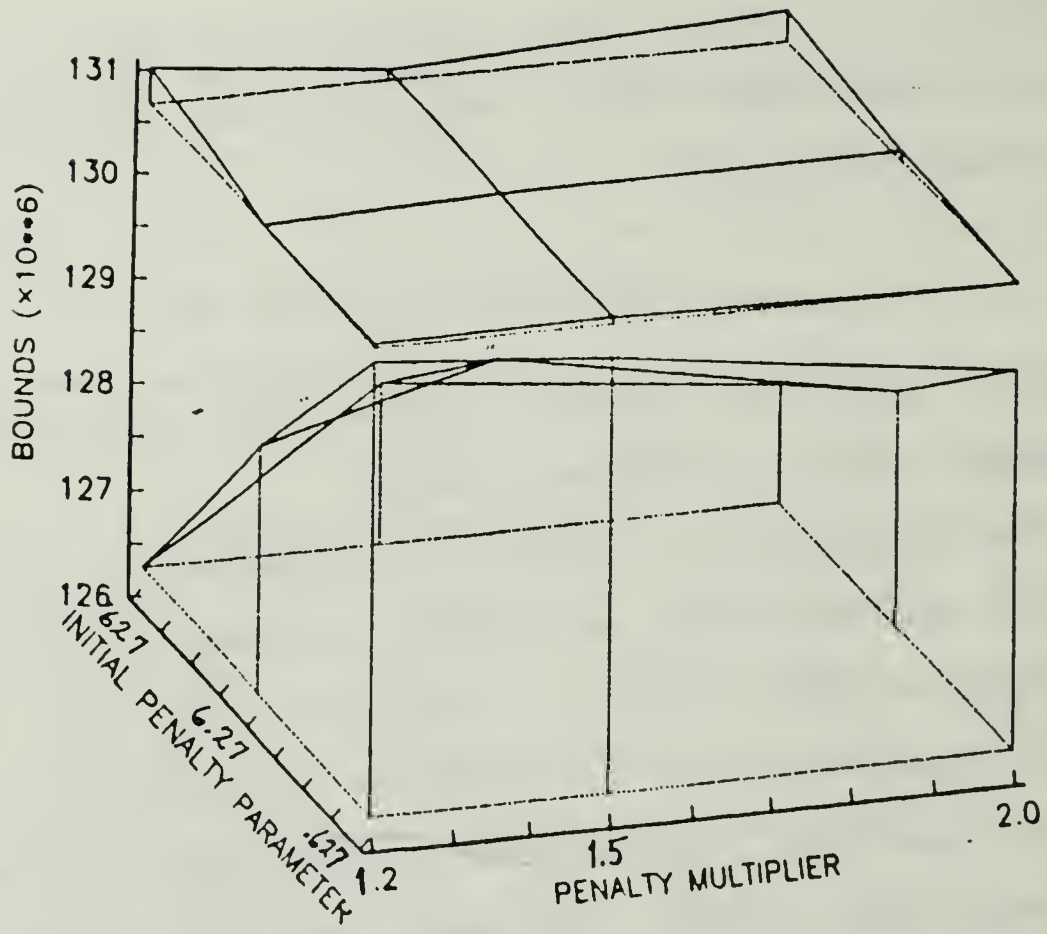

Figure 4.3 Response to Changing Parameters, RSD(A) on Problem 4H

solution until the set is bounded. In problem 10H, for instance, this requires 6 iterations.

One method to obtain early boundedness is to perform a resource allocation from the first relaxation, as in the RSD algorithms, and append this information to the DDC master problem as a "pseudocut." This method does provide an early feasible solution and the first pseudocut seems to remain binding for many iterations. However, our results show that it does not improve the point at which boundedness is naturally achieved in the master problem using cuts generated by the subproblems, nor does it affect the subsequent convergence trajectory.

Second, we must make an intelligent choice of the decomposition tolerance e in the cut thresholds ub+e. e acts as a relaxation parameter. If e is too small, we restrict the size of our improvements in the dual multipliers and inch uphill; if e is too large, we overshoot good multiplier choices and may oscillate through a sequence of extreme points which generate poor cuts and consequently poor dual solutions. Moderation is a virtue in the choice of e. For this set of test problems, we initially set e = 100,000 since no dual variable is expected to exceed 62,700 and obtain reasonable performance by reducing e by half whenever e-optimality is achieved (i.e., whenever the cuts cannot be satisfied). A final value e_f terminates this sequence of master problem relaxations.

Third, the decomposition goals are a daunting complication at first glance. However, a goal constraint for each master problem variable can be incorporated as a generalized upper bound, and the resulting master problem solved with very little additional effort.

Finally, we must consider the effects of accumulating too many cuts in the master problem. As the number of retained cuts grows, more time and space are required to solve the master, and more time is required to regenerate solutions from peripheral storage. However as the approximation to the dual response surface improves, many of the earlier cuts become non-binding, either temporarily or permanently. Figures 4.4 and 4.5 show cut histories for problems 4H and 10H, respectively. In these figures, the height of the "skyscraper" indicates the weight of the cuts at each iteration. Both results show that only a few cuts reenter a subsequent solution after once becoming slack, and then only at a small dual weight.

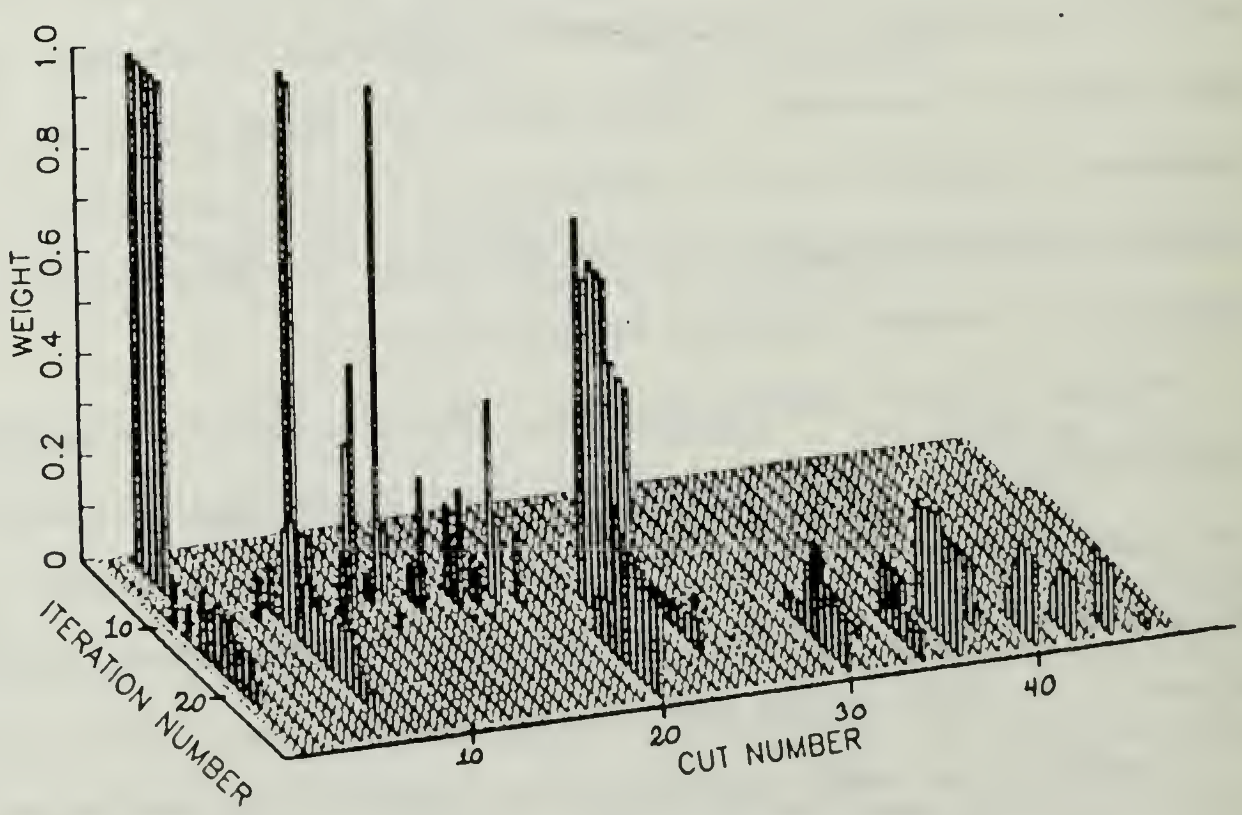

Figure 4.4 Active Cuts at Each Iteration
DDC on Problem 4H

We take advantage of this property by restricting the number of cuts retained to $n_{hold} = 20$. We also use an exponential moving average of the previous cut weightings (master problem dual variables) to determine which cuts to drop. In the (unlikely) event that a taut cut must be replaced, the (upper bound) value of a recoverable primal solution must be relaxed accordingly.

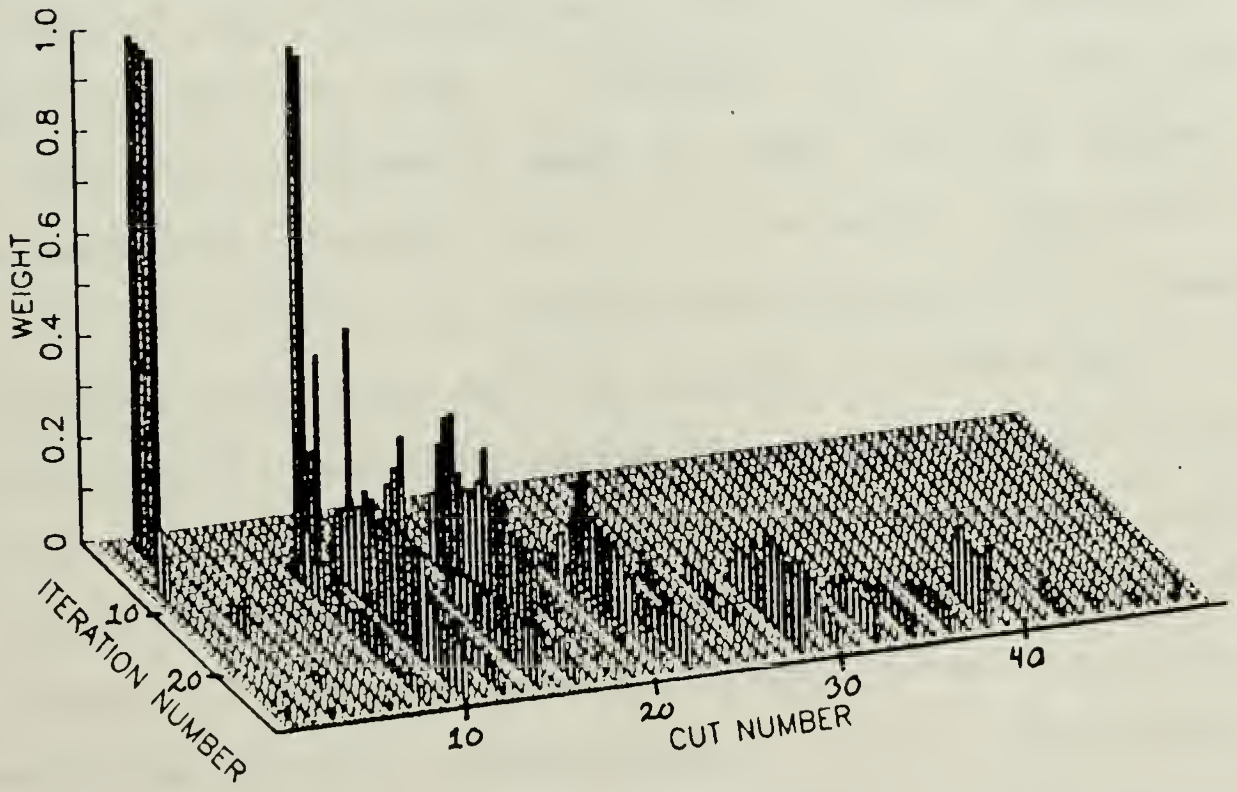

Figure 4.5 Active Cuts at Each Iteration
DDC on Problem 10H

The master problems are best solved by using a basis crash from the preceding solution in the sequence (except in the rare case that a taut cut has been replaced).

For our problems, the master problems have as many as $n_{hold} = 20$ (dense) cut constraints, and 1,071 decomposition goal (GUB) constraints, one for each of the 1,071 variables. Master problem solution times average 0.2 seconds (IBM 3033 AP).

The pure network subproblems have 3,300 nodes and 10,400 arcs, and solve in about 1 second per commodity. To reduce these dominating computation times, a hot start mechanism is used which initially restricts the network to those variables known to have had positive flow in any prior solution for that commodity. This restriction tends to reduce solution time as more experience is gained over iterations. After about 10 cuts, network subproblems rarely use any new arc not used before.

The networks subproblems are much more expensive to solve in DDC than the LP master problems. This is reversed from experience with primal decomposition, for which the master problems are commonly mixed integer LPs (e.g., see Geoffrion and Graves, 1974, or Brown, Graves and Honczarenko, 1983). Unfortunately, further mechanisms to reduce network solution times require additional storage, and we have limited our implementation to operating with a default 2 M-byte VM/CMS region. Overlays are used for each commodity subproblem and the master problem. We do not want to limit the number of commodities and have therefore used no data structure spanning commodities.

C. INTERMETHOD COMPARISONS

We now turn our attention to comparisons among the various algorithms. We establish acceptable solution gaps of 1% and 4% for the four and ten product problems, respectively, for two reasons. First, for such large-scale planning models, these levels of accuracy are adequate. Second, in order to maintain integer arithmetic in the network problems, we round multiplier estimates and capacity allocations, inducing the possibility that exact optima to the original problem have been excluded.

Figures 4.6 and 4.7 show the significant computational advantage of RSD(A) over RDLB. For both test problems 4H

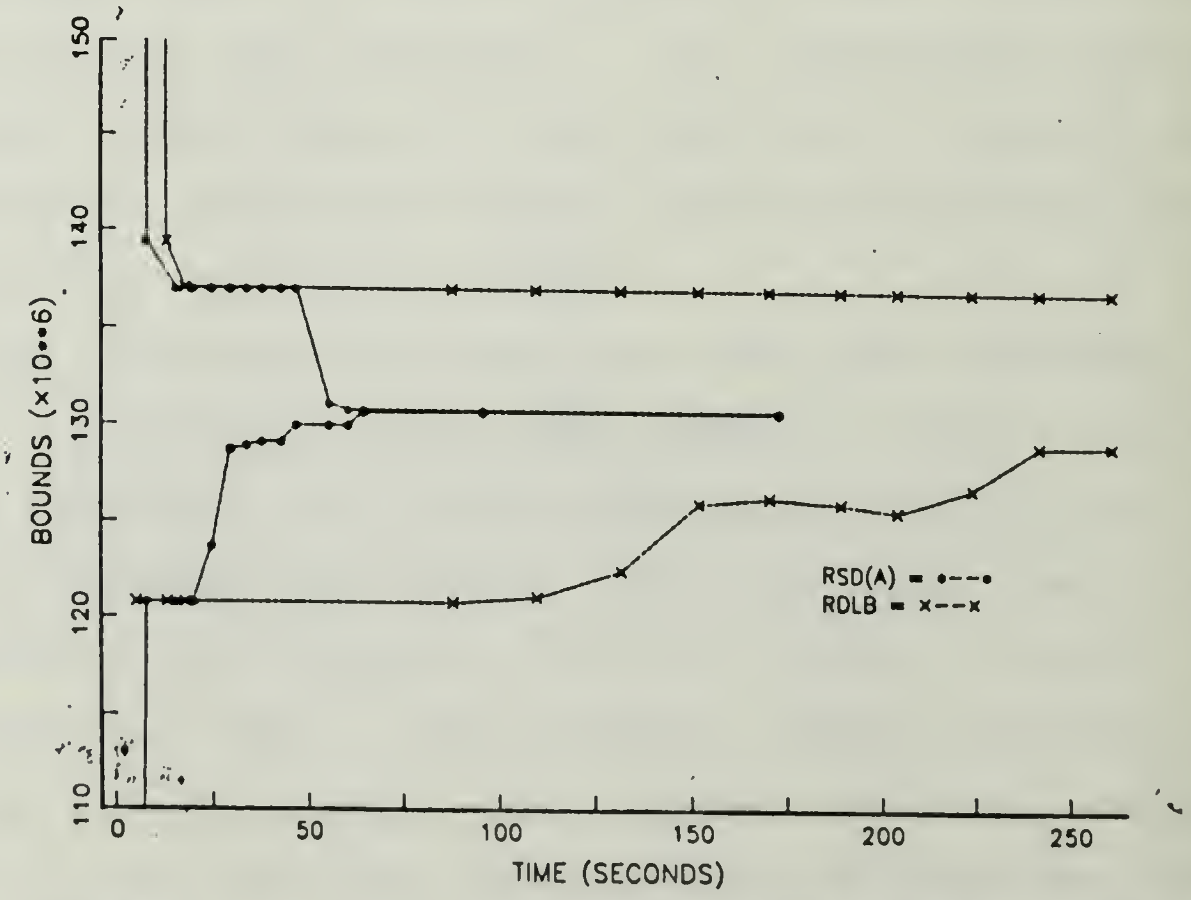

Figure 4.6 Solution Trajectory Comparison (Problem 4H) RSD(A) vs. RDLB

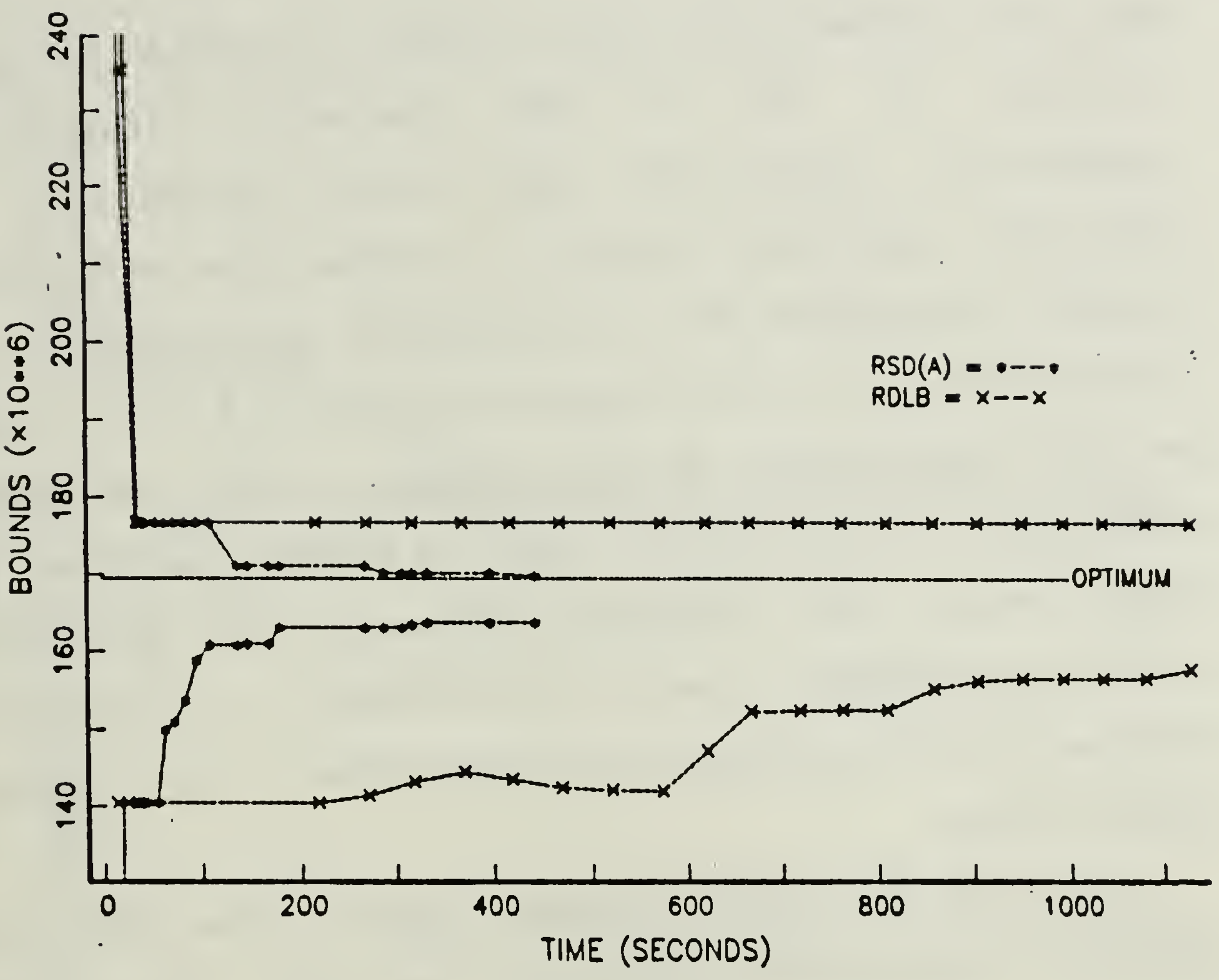

Figure 4.7 Solution Trajectory Comparison (Problem 10H) RSD(A) vs. RDLB

and 10H, the performance of the RDLB is much worse than RSD(A) because the subgradient reallocations are generally ineffective. In fact, although the subgradient reallocations lead to an optimal solution for problem 4E (not shown) no improvements are found for 10H, and improvements totaling .007% out of an 11.8% gap are made in 4H, only by using an exceedingly small step size.

The poor primal performance of RDLB is compounded by slow convergence of the Lagrangean problem. As indicated by the figures, it usually requires accumulation of three to five new subgradients before there is enough information available to find a good conjugate direction and substantially improve the lower bound. Since only a few costs are typically changed in solving the second set of network subproblems for the quadratic approximation to the line search, we use the previous bases as a "hot restart," but the time savings is not adequate to make the algorithm competitive in performance with the other algorithms. Even though other lower bounding strategies are available for testing, no further tests are performed on this algorithm because of its combined poor performance on both lower and upper bounds.

For the two test problems shown, the RDLB method requires significantly longer to do less. The algorithm terminates with final gaps of 5.83 and 10.7 percent, both unacceptable by our stated standard. The primal bound is relatively poor in both problems, corroborating the results of other researchers (e.g., Allen (1985)) which show subgradient-based resource direction unable to achieve an optimal solution.

We now compare RSD(A) to RSD(P) ($t = 1.5$). In Figure 4.8, RSD(A) takes about 60 seconds on problem 4H to reach a solution within .15% of optimal, while RSD(P) reaches a .69% solution in under 120 seconds. In Figure 4.9, the augmented form reaches a 4% solution to problem 10H in about 270 seconds and a 3.56% solution after 21 iterations in about

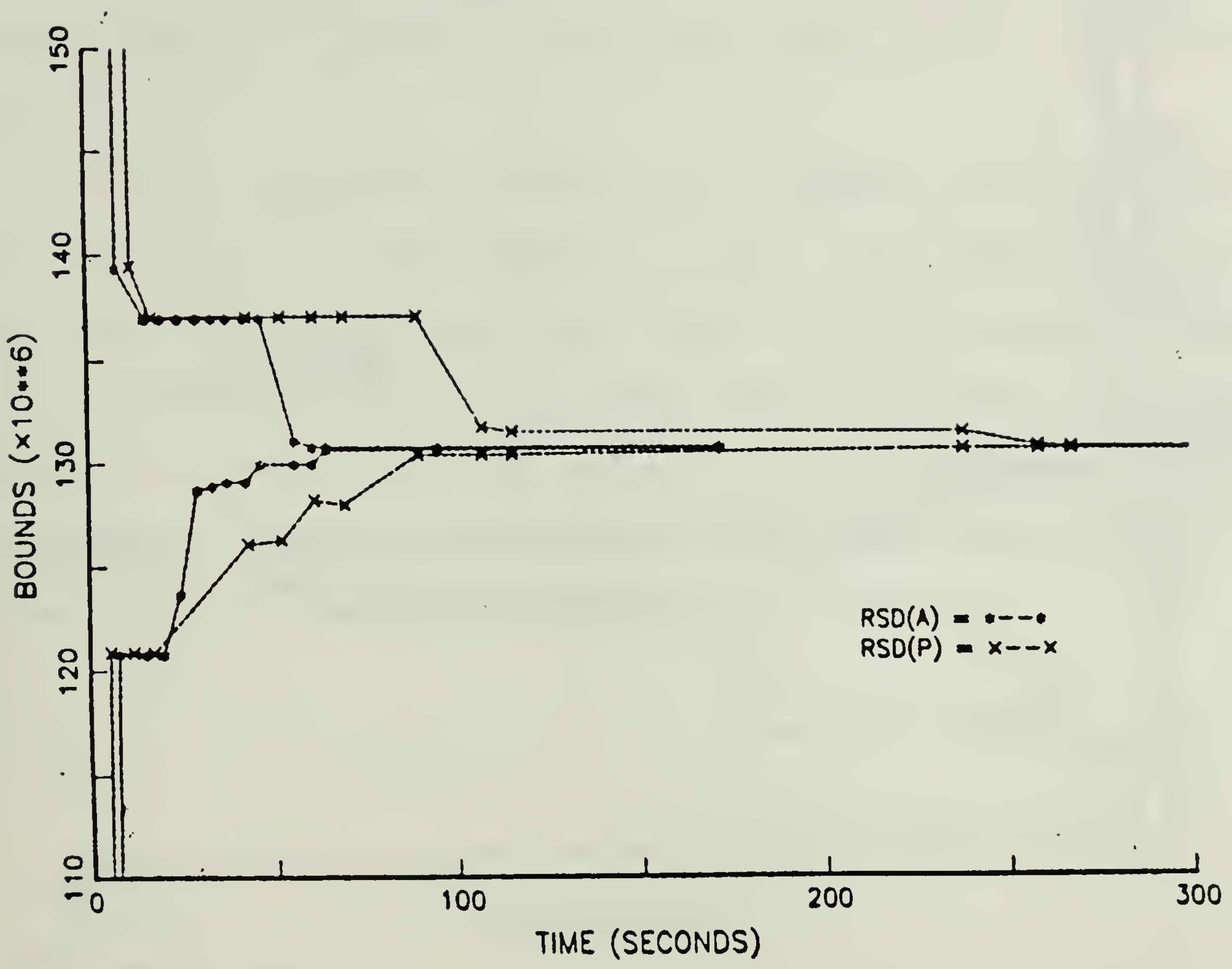

Figure 4.8 Solution Trajectory Comparison RSD(A) vs. RSD(P), Problem 4H

440 seconds. The penalty version takes nearly 470 seconds to reach the 4% level, closing at 3.93%. The X-system on an IBM 3081K solves this problem optimally in 1393.5 seconds using the network hot-start procedure (5,352 seconds without!). Considering the difference in computer speeds, RSD(A) reaches an acceptable solution in 1/10 the time of the X-system.

In comparing RSD(A) and RSD(P), we note not so much the difference in performance times as the movement of the lower

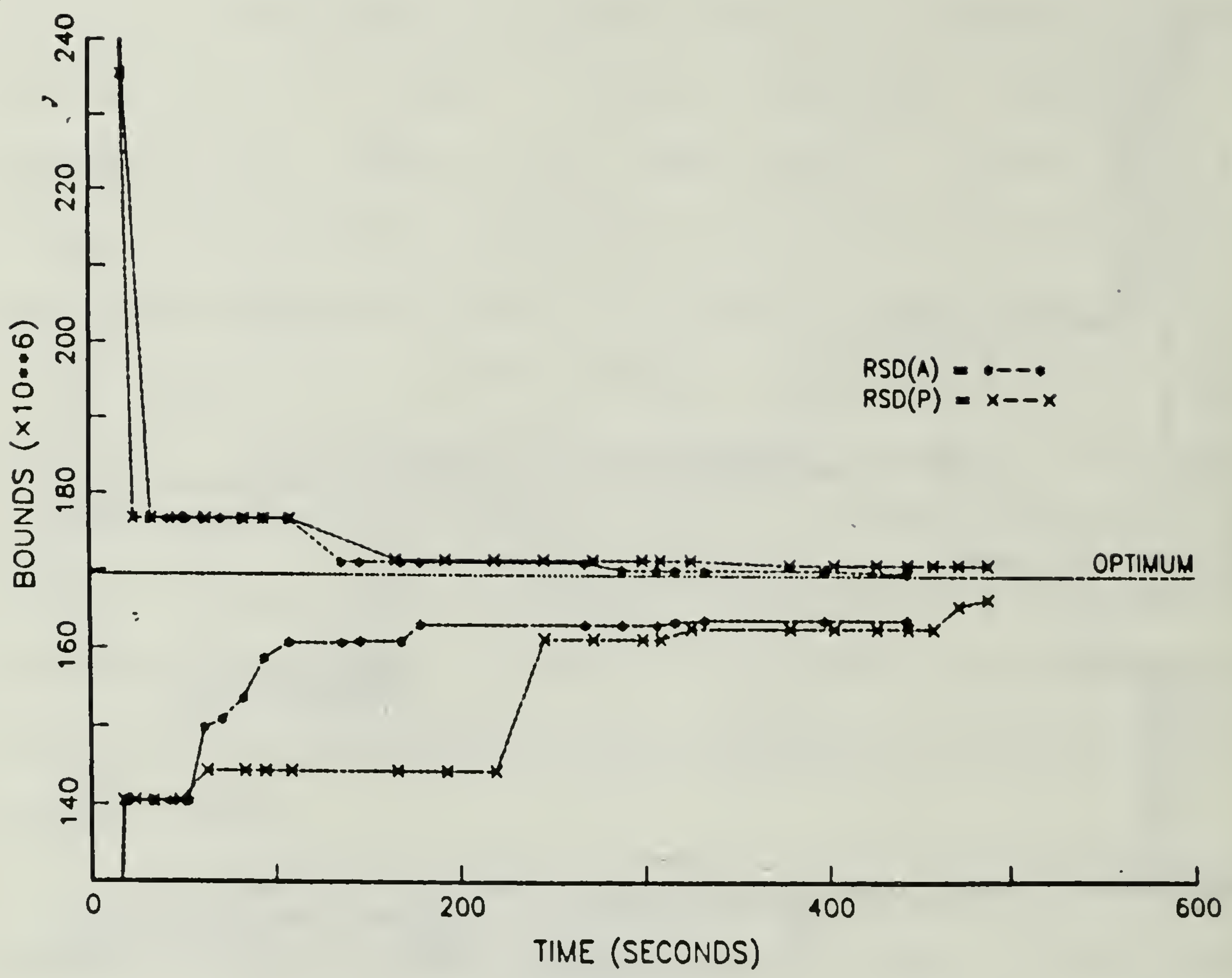

Figure 4.9 Solution Trajectory Comparisons (Problem 10H) RSD(A) vs. RSD(P)

bounds. The bound for RSD(A) climbs more quickly than RSD(P) apparently due to better instantaneous multiplier estimates and the fact that h is increased in smaller steps in RSD(A), resulting in a better-conditioned master problem at each iteration. Complete convergence of bounds is not anticipated for either algorithm, because we are rounding the multiplier estimates to perform integer arithmetic in the subproblems, we are not scaling the subproblems, and we are retaining only eight points. However, the solutions obtained by both algorithms are adequate, especially since the primal incumbents invariably seem to be very near optimal.

Finally, Figures 4.10 and 4.11 compare the performance of RSD(A) to the performance of DDC. For both problems 4H and 10H, the RSD(A) algorithm significantly outperforms the dual algorithm. While RSD takes about 60 seconds to reach a solution within .15% of optimal in 4H, DDC required 160 seconds to reach a 1% solution and about 180 seconds to reach a gap comparable to RSD(A).

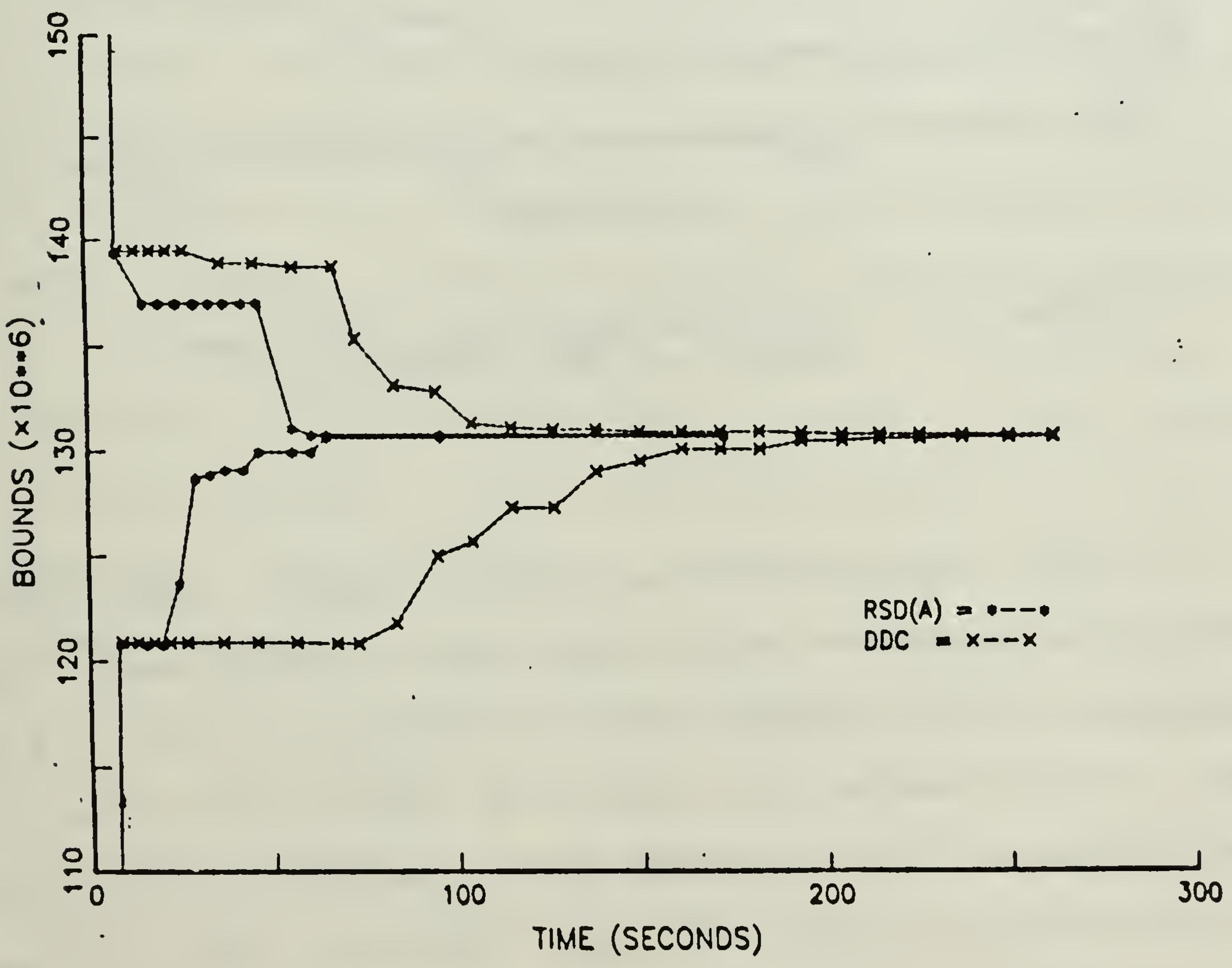

Figure 4.10 Solution Trajectory Comparison RSD(A) vs. DDC, Problem 4H

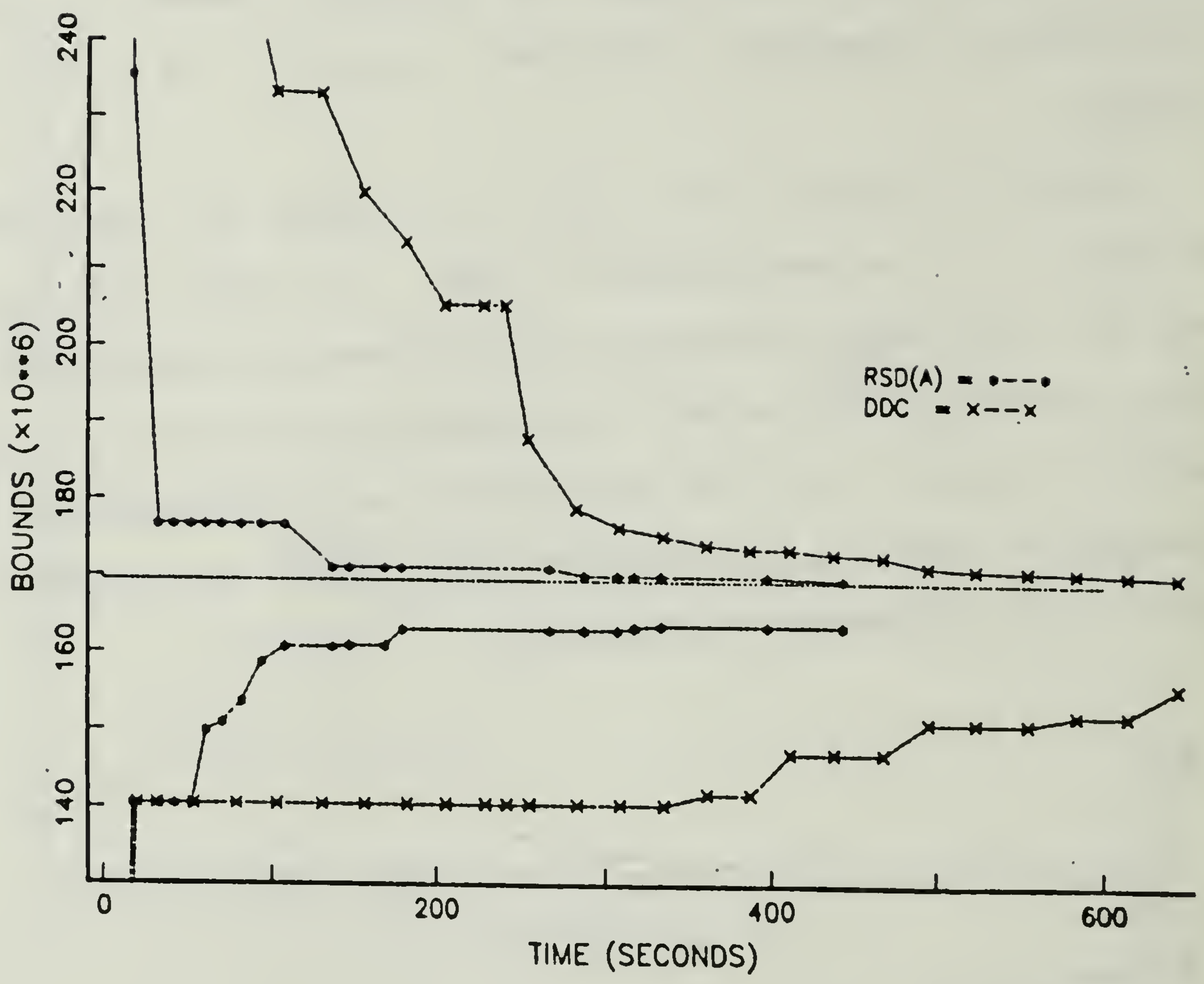

Figure 4.11 Solution Trajectory Comparison
RSD(A) vs. DDC, Problem 10H

In 10H, the comparison is similar. RSD(A) reaches a 4% solution in about 180 seconds and 3.56% in 440 seconds. DDC terminates at 650 seconds after 50 iterations with an 8.5% gap. These results are especially significant since the dual decomposition algorithm has a reputation for rapid initial progress. It is evident from both test problems that the RSD(A) algorithm performed better both on the upper and lower bounds.

Table 4.4 summarizes the final results for all algorithms and basic test problems run on an IBM 3033AP. Notice that RDLB runs reasonably well on 4E, but its times and solution quality are unacceptable for the hard problems. On the other hand, RSD(A) produces near-optimal primal solutions, acceptable bounds, and fast times for each test problem. The dual decomposition produces good final results, but through a poorer trajectory, and shows signs of laboring on problem 10H.

Finally, we construct a problem of 100 products to test DDC and RSD(A). It has 31 initial capacity violations with a maximum violation of 238% of arc capacity. Due to the increase in problem size, the initial value of h is reduced by a factor of 10 in RSD(A): no other changes have been made. Figure 4.12 shows that RSD(A) reaches an acceptable 4% solution in about 1000 seconds and concludes 21 iterations in 3000 seconds with a 1.5% gap. DDC terminates at 4090 seconds and 50 iterations with a 12.15% gap remaining. We note that a previous effort on this same problem using a resource-directive algorithm achieved an 11.8% solution in 1000 seconds (Staniec, 1984), but made no further improvements in an hour of computation. The RSD(A) algorithm achieved an acceptable solution in less than 17 minutes for a problem of some 330,000 constraints and 1,040,000 variables.

TABLE 4.4

SUMMARY DATA: COMPARATIVE PERFORMANCES

		RDLB				PSD(A)				DDC			
Prob.	Opt. Value	LB	UB	GAP	TIME	LB	UB	GAP	TIME	LB	UB	GAP	TIME
4E	340,916,981	340.6	340.92*	.08	171	340.75	340.92*	.048	71	340.45	341.01	.16	44.7
10E	367,135,103	357.75	369.45	3.17	114	364.8	367.22	.66	188	363.2	367.89	1.27	179.3
4H	130,739,585	129.0	137.0	5.83	234	130.66	130.74	.066	173	130.68	130.78	.07	268.5
10H	169,532,339	157.78	176.69	10.7	1028.9	165.8	169.9	2.4	501	156.1	170.6	8.5	646.4

All bounds x 10**6, Gaps in %, Times in seconds. *indicates an optimal primal solution

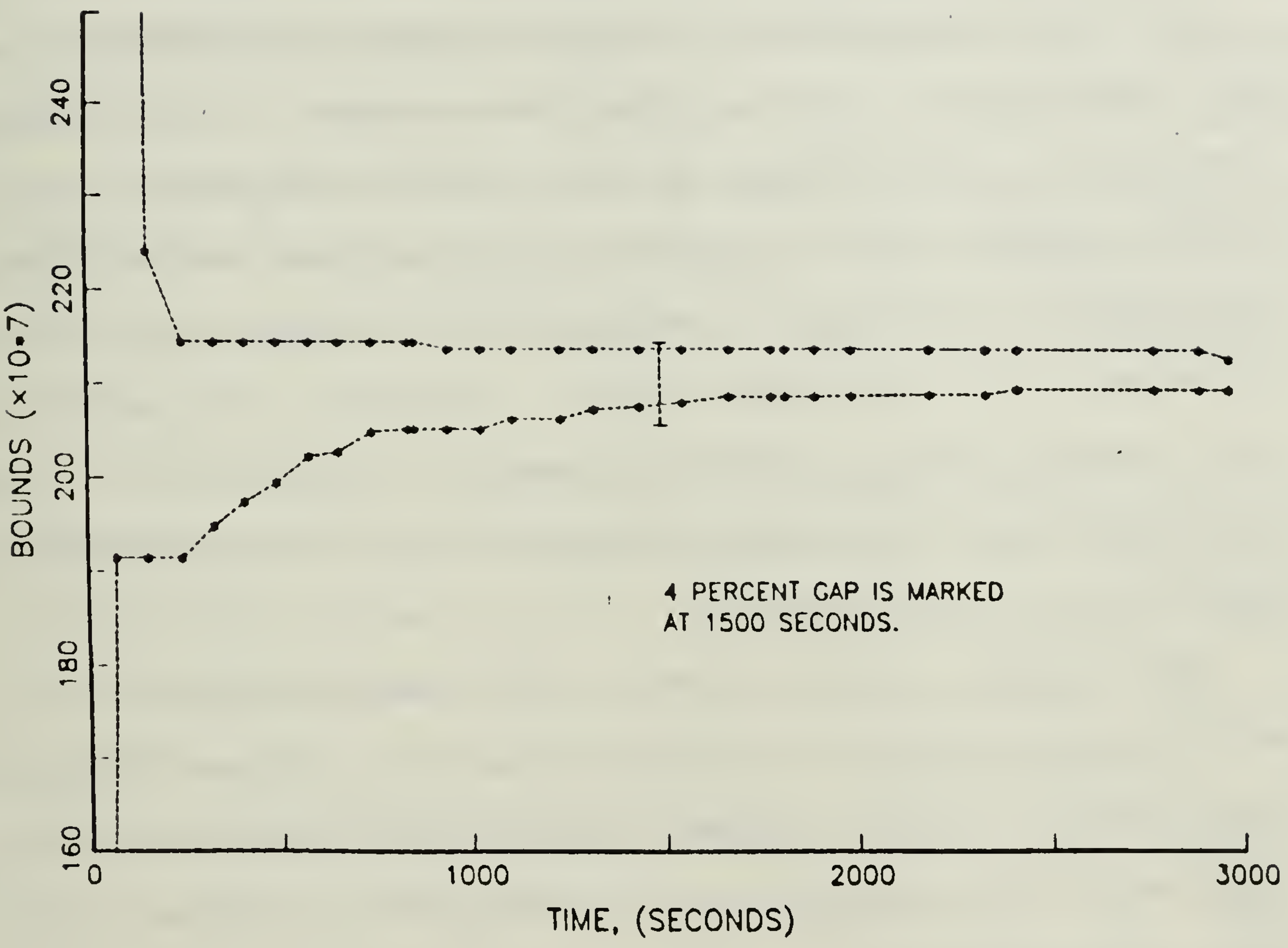

Figure 4.12 Solution Trajectory RSD(A) on 100 Commodities

V. CONCLUSION

We have presented three algorithms for solving large-scale linear programs that fall within the general framework of decomposition, with specific application to the MCTP.

The first algorithm, a resource-directive decomposition, has contributed a new, simplified method of projecting subgradient reallocations in the primal problems, and an improved method for terminating the algorithm via conjugate subgradient directions and approximate line searches in the Lagrangean lower bounding routine. However, the method is not computationally effective due to inability to find improving subgradient reallocations and due to computational burden in the line searches.

The second algorithm is a dual decomposition using a master problem which is the relaxed dual of the standard Dantzig-Wolfe master problem. Tests show the method to be computationally effective, but with slow convergence for very large problems.

The last algorithm (with two variants) is a new, non-linear price-directive decomposition using a penalty transformation of the original problem. Computational tests show the method to be approximately ten times faster than a direct simplex-based algorithm, and 2-3 times faster than the dual decomposition tested.

The initial success of the RSD algorithm prompts several areas for further investigation. First, the test suite is limited. We intend to test the algorithm on other large-scale problems

Second, Hearn et al. (1984) suggest that quadratic approximations to the master problem may speed its solution significantly without degrading convergence rate. Bertsekas (1982a) has described a projected Newton method for solving non-linear objectives in the presence of simple constraints. We propose development of a form of the Bertsekas algorithm which is capable of handling the penalty transformation of MCTP.

Third, it is reasonable to assume other non-linear objectives may have attractive features. For instance, the logarithmic barrier function has recently been the subject of extensive research for solving linear programs (see, for instance, Gill et al. (1986) or Karmarkar (1984)). We propose investigating interior point forms of the algorithm, with the possibility of developing a hybrid interior/exterior point algorithm, taking advantage of both penalty and barrier theory to solve linear programs.

Fourth, we propose a hybrid algorithm using the augmented Lagrangean form of RSD combined with dual decomposition to take advantage of the best properties of both algorithms. Via RSD, we quickly generate good estimates of the optimal dual multipliers, and therefore favorable extreme points. However, convergence slows due to the restricted number of extreme points retained. Increasing the extreme point count equates to increasing the time spent solving the master problem. However, these favorable extreme points may all be used to produce cuts for the dual decomposition master problem, which can be solved more quickly and precisely via linear programming. We anticipate the result to be a hybrid algorithm which has both favorable initial and terminal convergence properties.

Finally, we consider an extension of the algorithm from the shipment planning to a shipment scheduling framework, in which, for instance, we must make binary decisions on sea shipment arcs to account for shiploads of material. This will entail surrounding the RSD algorithm with a shell capable of interpreting information to make binary selections. One possible way to accomplish this is to develop a specialized version of the cross-decomposition algorithm of Van Roy (1986) which incorporates RSD.

In conclusion, the result of this research has been a new, computationally attractive algorithm for large scale linear programs using non-linear methods. Also, through the computational results produced, it has documented some of the shortcomings of the subgradient approach in resource direction. It is evident that more information (e.g., a formal Benders master problem) is required to solve difficult problems by resource direction.

APPENDIX

CONVEXITY OF THE QUADRATIC PENALTY FUNCTION

The following lemma demonstrates that the quadratic penalty form of the MCTP is everywhere convex, which makes it appropriate for the application of RSD. Convexity of other penalty forms may be proved in a similar manner.

Lemma: The function, $q(h,x)$ is convex for all $x \quad F = \{x | Nx = b_2, 0 \leq x \leq b_1\}$.

Proof: $q(h,x) = cx + P(h,x)$, where $P(h,x) = \frac{1}{2}h[(Ax-b_1)^+]'(Ax-b_1)^+$. Since cx is trivially convex and the sum of convex functions is convex, we need only show $P(h,x)$ to be convex; that is, that

$$P(h,x) \geq P(h,\bar{x}) + \nabla_x P(h,\bar{x})(x-\bar{x}), \quad \text{for all } x \in F$$

where $\nabla P_x(h,x) = h[(Ax-b_1)^+]'A$

Then we define $J_V = \{j | A_j\overline{x-b}_{1j} > 0\}$ and revert to summation form. $P(h,x)$ is convex if

$$\sum_j \frac{1}{2}h[(A_jx-b_{1j})^+]^2 \geq \sum_{j\in J_V} \frac{1}{2}h(A_j\overline{x-b}_{1j})^2 + \sum_{j\in J_V} h(A_j\bar{x}-b_{1j})A_j(x-\bar{x}).$$

Rewriting the left side as

$$h[(A_j x-b_1{}^j)^+]^2 = \sum_{j\in J_V} h[(A_j x-b_{1j})^+]^2 + \sum_{j\notin J_V} h[A_j x-b_{1j})^+]^2$$

and noting that $\sum_{j\notin J_V} h[(A_j x-b_{1j})^+]^2 \geq 0$ for all x

we may prove convexity if

$$\sum_{j\in J_V} \tfrac{1}{2}h[(A_j x-b_{1j})^+]^2 \geq \sum_{j\in J_V} \{\tfrac{1}{2}h(A_j\bar{x}-b_{1j})^2 + h(A_j\bar{x}-b_{1j})A_j(x-\bar{x})\}$$

or, considering additivity of convex functions, if

$$\frac{1}{2}h[(A_j x-b_{1j})^+]^2 \geq \frac{1}{2}h[(A_j\bar{x}-b_{1j})^+]^2 + h(A_j\bar{x}-b_{1j})^+A_j(x-\bar{x})$$

for all $x \in F$, and $j \in J_V$.

If $(A_j\bar{x}-b_{1j})^+ = 0$, the right side of the inequality is 0, and $h[(A_j x-b_{1j})^+]^2 \geq 0$ trivially. Now, for $(A_j\bar{x}-b_{1j})^+ > 0$, letting $\bar{u}_j = A_j\bar{x}-b_{1j}$, dropping h we write

$$\tfrac{1}{2}(A_j\bar{x}-b_{1j})^2 + (A_j\bar{x}-b_{1j})A_j(x-\bar{x}) = \tfrac{1}{2}\bar{u}_j(A_j\bar{x}-b_{1j}) + \bar{u}_j(A_j x-A_j\bar{x})$$

$$= \bar{u}_j[A_j x-(b_{1j})+\tfrac{1}{2}(A_j\bar{x}-b_1)-A_j\bar{x}+(b_{1j})]$$

$$= \bar{u}_j[A_j x-b_{1j} - \tfrac{1}{2}(A_j\bar{x}-b_{1j})]$$

$$= \bar{u}_j(A_j x-b_{1j}) - \tfrac{1}{2}\bar{u}_j{}^2$$

Convexity is proved if $\frac{1}{2}[(A_j x-b_{1j})^+]^2 \geq u_j(A_j x-b_{1j}) - \frac{1}{2}u_j{}^2$. Two cases occur:

For all $x \mid (A_j x - b_{1j})^+ = 0$, then $A_j x - b_{1j} \le 0$ and we have $\frac{1}{2}((A_j x - b_{1j})^+)^2 = 0 \ge \bar{u}_j (A_j x - b_{1j}) - \frac{1}{2}\bar{u}_j{}^2 =$ $\{(A_j x - b_{1j})^+]^2 + (A_j \bar{x} - b_{1j})^+ A_j (x - \bar{x})$.

For all $x \mid (A_j x - b_{1j})^+ > 0$, we let $\hat{u}_j = (A_j x - b_{1j})$ and note $(\hat{u}_j - \bar{u}_j)^2 \ge 0$.

But $(\hat{u}_j - u_j)^2 = \hat{u}_j{}^2 - 2\hat{u}_j u_j + u_j{}^2 \ge 0$, and $\hat{u}_j{}^2 \ge -u_j{}^2 + 2\hat{u}_j - u_j$, so multiplying by h and restoring values, we have

$$\frac{1}{2} h[(A_j x - b_{1j})^+]^2 \ge \frac{1}{2} h[(A_j \bar{x} - b_{1j})^+]^2 + h(A_j \bar{x} - b_{1j})^+ A_j (x - \bar{x})$$

Since $P_j(h,x)$ is convex for all j and the summation of convex functions is convex, $q(h,x)$ is convex. QED

LIST OF REFERENCES

Agmon, S., "The Relaxation Method for Linear Inequalities," Canadian Journal of Mathematics, Vol. 6, 1954, pp. 382-392.

Ali, I., Barnett, D., Farhangian, K., Kennington, J., McCarl, B., Patty, B., Shetty, B., and Wong, P., "Multicommodity Network Problems: Applications and Computations," IIE Transactions, V. 16-2, 1984, pp. 127-134.

Ali, I., R.V. Helgason, and J.L. Kennington, The Convex Cost Network Flow Problem: A Survey of Algorithms, Technical Report ORGM 78001, Southern Methodist University, 1978.

Ali, A.I., Helgason, R.V., Kennington, J.L., and Lall, H., "Primal-Simplex Network Codes: State-of-the-Art Implementation Technology, Networks, Vol. 8, 1978, pp. 315-339.

Ali, A., R. Helgason, J. Kennington, and H. Lall, "Computational Comparison among Three Multicommodity Network Flow Algorithms," Operations Research, Vol 28-4, 1980, pp. 995-1000.

Allen, E., "Using Two Sequences of Pure Network Problems to Solve the Multicommodity Network Flow Problem," unpublished dissertation, Southern Methodist University, April 1985.

Assad, A., "Multicommodity Network Flows--A Survey," Networks, Vol. 8, 1978, pp. 37-91.

Barr, R.S., Glover, F., and Klingman, D., "Enchancements of Spanning Tree Labelling Procedures for Network Optimization," INFOR, Vol. 17-1, 1979, pp. 16-34.

Bazaraa, M. and J. Jarvis, Linear Programming and Network Flows, John Wiley and Sons, New York, 1977.

Bazaraa, M.S. and H.D. Sherali, "On the Choice of Step Size in Subgradient Optimization," European Journal of Operational Research, Vol. 7, 1981, pp. 380-388.

Bazaraa, M.S. and C.M. Shetty, Nonlinear Programming, Theory and Algorithms, John Wiley, New York, 1979.

Benders, J.F., "Partitioning Procedure for Solving Mixed-Variables Programming Problems," Numerische Mathematik, Vol. 4, 1962, pp. 238-252.

Bertsekas, D.P., "Projected Newton Methods for Optimization Problems with Simple Constraints," SIAM Journal of Control and Optimization, V. 20-2, 1982a, pp. 221-296.

Bertsekas, D., Constrained Optimatization and LaGrange Multiplier Methods, Academic Press, New York, 1982b.

Bradley, G., G.G. Brown, and G. Graves, "Design and Implementation of Large-Scale Primal Transshipment Algorithms," Management Science, Vol. 24-1, 1977, pp. 1-34.

Brown, G.G. and G.W. Graves, XS Mathematical Programming System, perpetual working paper, 1987.

Brown, G.G., G.W. Graves, and M.D. Honczarenko, "Design and Operation of a Multicommodity Production-Distribution System Using Primal Goal Decomposition," Naval Postgraduate School report NPS55-83-010, Monterey, California, May 1983.

Cremeans, J.E., R.A. Smith, and G.R. Tyndall, "Optimal Multicommodity Network Flows with Resource Allocation," Naval Research Logistics Quarterly, V. 17, 1970, pp. 269-280.

Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, NJ, 1963.

Dantzig, G.B., and R.M. Van Slyke, "Generalized Upper Bounding Techniques," Journal of Computer and System Sciences, Vol. 1, 1967, pp. 213-226.

Dantzig, G.B., and P. Wolfe, "Decomposition Principle for Linear Programs," Operations Research, Vol. 8, 1960, pp. 101-111.

Demjanov, V., "Algorithms for Some Minimax Problems," Journal of Computer and System Sciences, Vol. 2, 1968, pp. 342-380.

Evans, J.R., "The Simplex Method for Integral Multicommodity Networks," Naval Research Logistics Quarterly, V. 25, 1978, pp. 31-38.

Evans, J.R., "A Combinatorial Equivalence Between a Class of Multicommodity Flow Problems and the Capacitated Transportation Problem," Mathematical Programming, Vol. 10, 1976, pp. 401-404.

Evans, J.R., "On Equivalent Representations of Certain Multicommodity Networks as Single Commodity Flow Problems," Mathematical Programming, Vol. 15, 1978, pp. 92-99.

Evans, J.R., "A Network Decomposition/Aggregation Procedure for a Class of Multicommodity Transportation Problems," Networks, Vol. 13, 1983, pp. 197-205.

Evans, J.R., J.J. Jarvis, and R. Duke, "Graphic Matroids and the Multicommodity Transportation Problem," Mathematical Programming, Vol. 13, 1977, pp. 323-328.

Fisher, Marshall, "An Application's Oriented Guide to Lagrangean Relaxation," Interfaces, Vol. 15-2, 1985, pp. 10-21.

Ford, L. and D. Fulkerson, "A Suggested Computation for Maximal Multicommodity Network Flows," Management Science, Vol. 5, 1958, pp. 97-101.

Ford, L.R., and D.R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, NJ, 1962.

Frank, M. and P. Wolfe, "An Algorithm for Quadratic Programming," Naval Research Logistics Quarterly, Vol. 3, 1956, pp. 95-110.

Fulkerson, D.R., "An Out-of-Kilter Method for Minimal-Cost Flow Problems," Journal of the Society of Industrial and Applied Mathematics, Vol. 9-1, 1961, pp. 18-27.

Fulkerson, D.R., and G.B. Dantzig, "Computation of Maximal Flows in Networks," Naval Research Logistics Quarterly, Vol. 2-4, 1955, pp. 277-283.

Geoffrion, A.M., "Elements of Large-Scale Mathematical Programming," Rand Report R-481-PR, 1969.

Geoffrion, A.M., "Primal Resource-Directive Approaches for Optimizing Nonlinear Decomposable Systems," Operations Research, Vol. 18, 1970, pp. 375-403.

Geoffrion, A.M., and G.W. Graves, "Multicommodity Distribution System Design by Bender's Decomposition," Management Science, Vol. 29-5, 1974, pp. 822-844.

Gerald, C.F., and P.O. Wheatley, Applied Numerical Analysis, 3rd ed., Addison Wesley, Menlo Park, California, 1984.

Gill, P.E., W. Murray, M.A. Saunders, M.H. Wright, "A Note on Nonlinear Approaches to Linear Programming," Systems Optimization Laboratory Technical Report SOL 86-7, Stanford University, April 1986.

Glover, F., Karney, D., and Klingman, D., "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," Networks, Vol. 4-3, 1974, pp. 191-212.

Glover, F., Karney, D., Klingman, D., and Napier, A., "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," Management Science, Vol. 20-5, 1974, pp. 793-813.

Glover, F., Klingman, D., and Stutz, J., "Augmented Threaded Index Method for Network Optimization," INFOR, Vol. 12-3, 1974, pp. 293-298.

Goffin, J.L., "On Convergence Rates of Subgradient Optimization Methods," Mathematical Programming, Vol. 13, 1977, pp. 329-347.

Graves, G. and R. McBride, "The Factorization Approach to Large-Scale Linear Programming," Mathematical Programming, Vol. 10, 1976, pp. 91-110.

Graves, G.W., and Van Roy, T.J., "Decomposition for Large Scale Linear and Mixed Integer Linear Programming," UCLA Technical Report, 1979.

Grigoriadis, M.D., and White, W.W., "A Partitioning Algorithm for the Multicommodity Network Flow Problem," Mathematical Programming, Vol. 3, 1972, pp. 157-177.

Hartman, J.K. and L-S. Lasdon, "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Algorithms," Networks, Vol. 1, 1972, pp. 333-354.

Hearn, D.W., S. Lawphongpanich, and J.A. Ventura, "Restricted Simplicial Decomposition: Computation and Extensions," Research Report 84-38, University of Florida, Gainesville, 1984.

Hearn, D.W., S. Laephongpanich, and J.A. Ventura, "Finiteness in Restricted Simplicial Decomposition," Operations Research Letters, Vol. 4-3, 1985, pp. 125-130.

Held, M. and R. Karp, "The Travelling-Salesman Problem and Minimum Spanning Trees," Operations Research, Vol. 18, 1970, pp. 1138-1162.

Held, M., and R. Karp, "The Travelling Salesman Problem and Minimum Spanning Trees: Part II," Mathematical Programming, Vol. 1, 1971, pp. 6-25.

Held, M., P. Wolfe, and H.P. Crowder, "Validation of Subgradient Optimization," Mathematical Programming, Vol. 6, 1974, pp. 62-88.

Helgason, R.V., and Kennington, J.L., "A Product Form Representation of the Inverse of a Multicommodity Cycle Matrix," Networks, Vol. 7, 1977, pp. 297-322.

Hestenes, M.R., Optimization Theory: The Finite Dimensional Case, John Wiley and Sons, New York, 1975.

Ho, J.K., "Convergence Behavior of Decomposition Algorithms for Linear Programs," Operations Research Letters, Vol. 3, 1984a, pp. 91-94.

Ho, J.K., "Recent Advances in the Decomposition Approach to Linear Programming," lecture presented at the NATO Advanced Study Institute on Computational Mathematical Programming, Bad Windsheim, West Germany, July 1984b.

Ho, J.K. and E. Loute, "Computational Experience with Advanced Implementation of Decomposition Algorithms for Linear Programming," Mathematical Programming, Vol. 27, 1983, pp. 283-290.

Johnson, E.L., "Networks and Basic Solutions," Operations Research, Vol. 14-4, 1966, pp. 619-623.

Karmarkar, N.K., "A New Polynomial-Time Algorithm for Linear Programming," AT&T Bell Laboratories Technical Memorandum TM 11216-840126-04, 1984.

Kelley, J.E., "The Cutting Plane Method for Solving Convex Programs," SIAM J. Appl. Math, Vol. 8, 1960, pp. 703-712.

Kennington, J., "Solving Multicommodity Transportation Problems Using a Primal Partitioning Simplex Technique," Naval Research Logistics Quarterly, Vol. 24, 1977, pp. 309-325.

Kennington, J., "A Survey of Linear Cost Multicommodity Network Flows," Operations Research, Vol. 26, 1978, pp. 209-236.

Kennington, J., and Helgason, R., Algorithms for Network Programming, John Wiley & Sons, New York, NY, 1980.

Kennington, J. and M. Shalaby, "An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems," Management Science, Vol. 23-9, 1977, pp. 994-1004.

Kuhn, H.W., "The Hungarian Method for the Assignment Problem," Naval Research Logistics Quarterly, Vol. 2, 1955, pp. 83-97.

Lasdon, L.S., Optimization Theory for Large Systems, MacMillan, New York, 1970.

Luenberger, D.G., Linear and Nonlinear Programming, 2nd ed., Addison-Wesley, Menlo Park, California, 1984.

Meyer, G.G., "Accelerated Frank-Wolfe Algorithms," SIAM Journal on Control, Vol. 12-4, 1974, pp. 655-663.

Motzkin, T.S. and I.J. Schoenberg, "The Relaxation Method for Linear Inequalities," Canadian Journal of Mathematics, Vol. 6, 1954, pp. 393-404.

Poljak, B.T., "A General Method of Solving Extremum Problems," Soviet Mathematical Doklady, Vol. 8-3, 1967, pp. 593-597.

Rockafellar, R.T., Convex Analysis, Princeton University Press, 1970.

Rosenthal, R., "Multicommodity Network Flow Optimization," NRC Research Proposal, University of Tennessee, 1983.

Sandi, C., "Subgradient Optimization," Chapter 3 in Combinatorial Optimization, ed. N. Christofides, John Wiley & Sons, New York, NY, 1979.

Srinivasan, V., and Thompson, G.L., "Accelerated Algorithms for Labelling and Relabeling of Trees, with Applications to Distribution Problems," Journal of the Association for Computing Machinery, Vol. 19-4, 1972, pp. 712-726.

Srinivasan, V., and Thompson, G.L., "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," Journal of the Association for Computing Machinery, Vol. 20, 1973, pp. 194-213, 1973.

Staniec, C.J., Design and Solution of an Ammunition Distribution Model by a Resource-Directive Multicommodity Network Flow Algorithm, Master's Thesis, Naval Postgraduate School, Monterey, CA, 1984.

Swoveland, C., "A Two-Stage Decomposition Algorithm for a Generalized Multicommodity Flow Problem," INFOR, Vol. 11, 1973, pp. 232-244.

Tomlin, J.A., "Minimum Cost Multicommodity Network Flows," Operations Research, Vol. 14-1, 1966, pp. 45-51.

Van Roy, Tony J., "A Cross Decomposition Algorithm for Capacitated Facility Location," Operations Research, Vol. 34-1, 1986, pp. 145-163.

von Hohenbalken, B., "Simplicial Decomposition in Nonlinear Programming Algorithms," Mathematical Programming, Vol. 13, 1977, pp. 49-68.

Weigel, H.S. and J.E. Cremeans, "The Multicommodity Network Flow Model Revised to Include Vehicle Per Time Period and Node Constraints," Naval Research Logistics Quarterly, Vol. 19, 1972, pp. 77-89.

Wolfe, P., "Convergence Theory in Nonlinear Programming," Chapter 1, in Integer and Nonlinear Programming, ed., J. Abadie, American Elsevier, New York, 1970.

Wolfe, P., "A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions," Mathematical Programming Study 3, 1975, pp. 145-173.

Wollmer, R.D., "Multicommodity Networks with Resource Constraints: The Generalized Multicommodity Flow Problems," Networks, Vol. 1, 1972, pp. 245-263.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor Gerald G. Brown, Code 55Bw Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	2
4.	Major Cyrus J. Staniec, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93943-5000	1